Hierarchical Heuristic approaches for Path Finding

16.12.2014

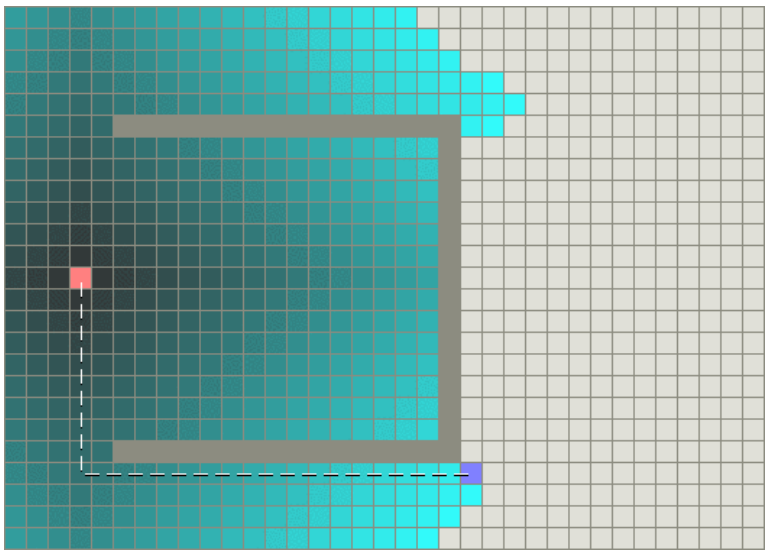
Lorenzo Di Giuseppe - 227515
Sistemi di elaborazione dell'informazione

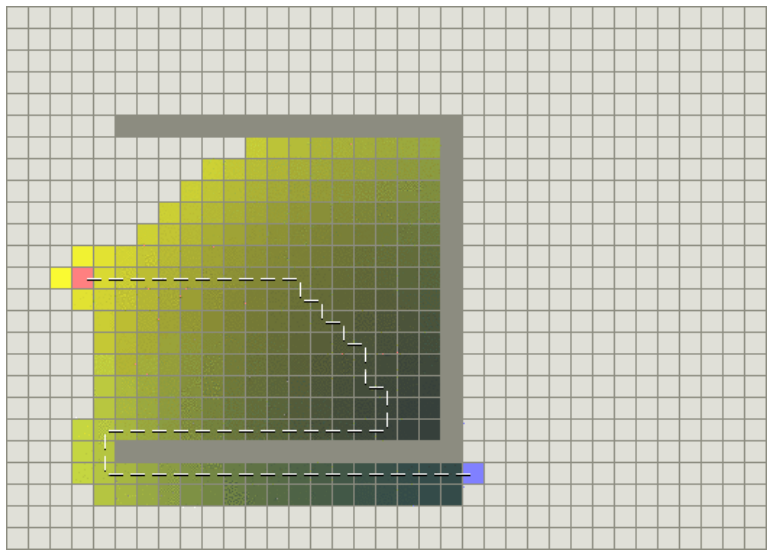
Introduction

- Path finding is a problem in many context
- The paths have to satisfy constraints related to applications
  - Robots path have to be kinematically feasible
  - Intelligent transportation systems requires traffic re-routing
  - Video-games intelligence requires real time response
  - ...
- Different applications usually take to different solutions

Single Pair Shortest Path

- How is it possible to find a path between two points?
- Possible solutions:
  - Computing paths from the source with Dijkstra taking the one that carries to destination
    - It's too expensive on large networks
  - Using estimations of distance from source composing a path to the arrival as in Best-First Search
    - Paths are not so short





Index

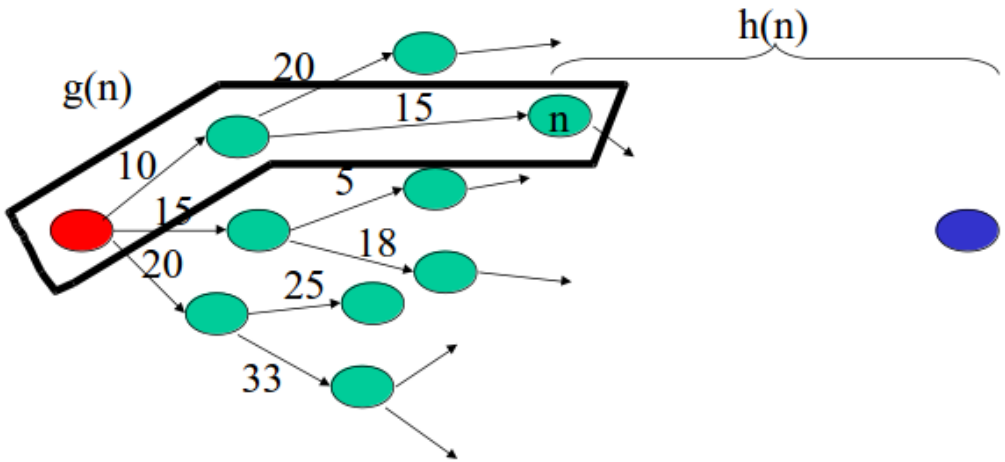
These topics are introduced in this presentation:

1. A*
   - Peter Hart, Nils Nilsson and Bertram Raphael
2. A Hierarchical Path Planning Approach Based on Reinforcement Learning for Mobile Robots
   - Qi Guo, Lei Zuo, Rui Zheng and Xin Xu
3. A Hierarchical Path View Model for Path Finding in Intelligent Transportation Systems
   - Yun-Wu Huang, Ning Jing and Elke A. Rundensteiner

A*

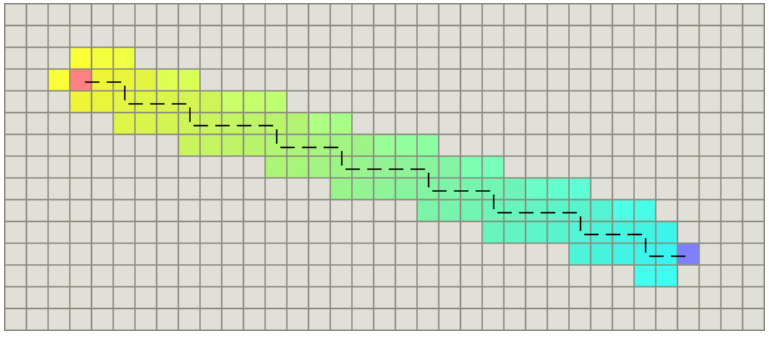
"A is one of the most used algorithm for path finding thanks to its flexibility and work field."*

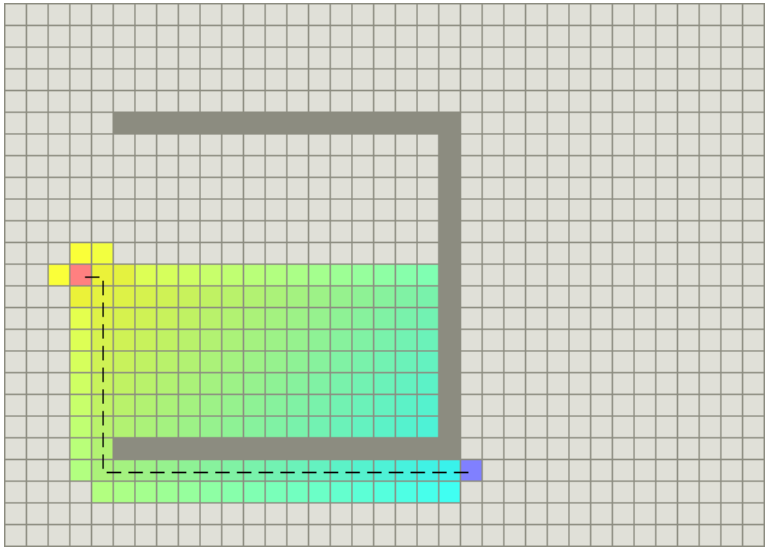
- A* mixes the Dijkstra principle with Greedy Best-First-Search
- It expands first nodes closer to the source that are also estimated closer to the arrival
  - It's a priority visit that prefers nodes with minimum $g(n) + h(n)$
  - The estimation $h(n)$ is done with an heuristic function
    - e.g. Euclidean distance



A*

- Heuristic controls the behaviour of the algorithm
  - If $h(n) = const$, A* acts like Dijkstra
  - Else the found path may not be the shortest but the algorithm finds it quickly
- A* performs less expansions than Dijkstra and finds better path than BFS

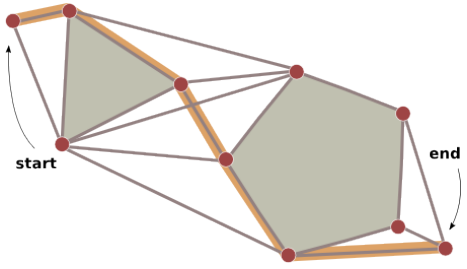




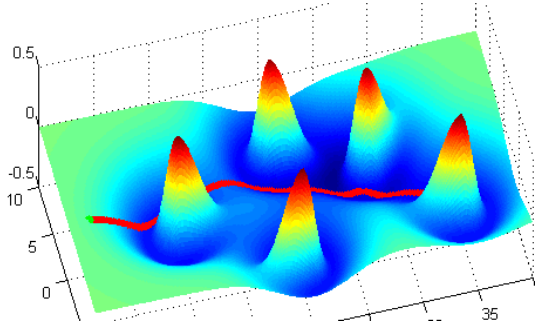
Path Finding in robotic

"The main problem of path finding is the kinematic feasibility of the path and the collision avoidance"

- A* paths could be kinematically infeasible

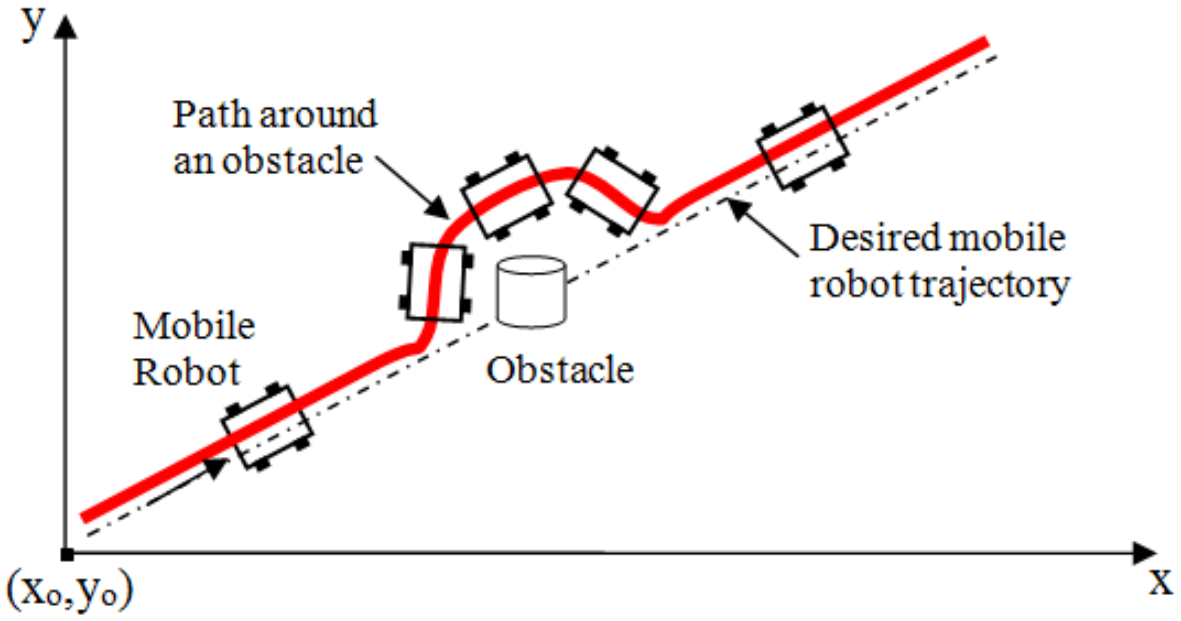


- Potential Field Method
  - Paths are kinematically feasible
  - Destination is sometimes unreachable



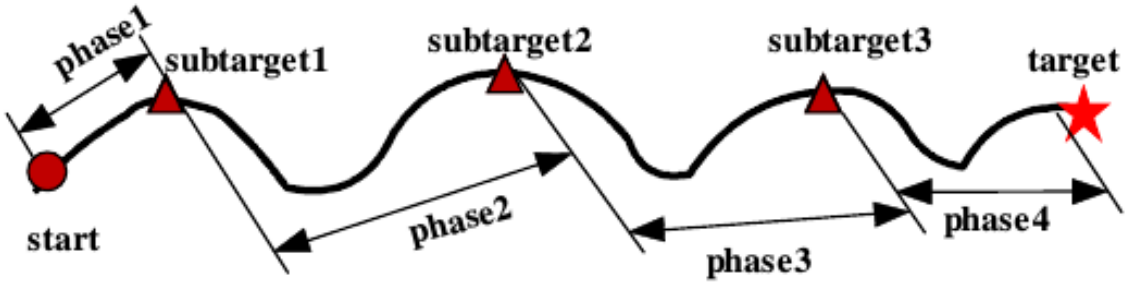
A HPP Approach Based on RL for Mobile Robots

- Path planning is done in two independent layers
  - The first (higher-level) layer computes shortest path to a desired point
  - The second (lower-level) layer finds a kinematically feasible path around the computed one



First Layer: A*

- A* is used to search a short, obstacle-free geometric path
  - It's possible to use other algorithms instead A* or metrics to satisfy different vehicular tasks:
    - e.g. a path that minimizes fuel consumptions
- The found path is sampled thinking at the kinematic constraints to determine sub-targets for the other level



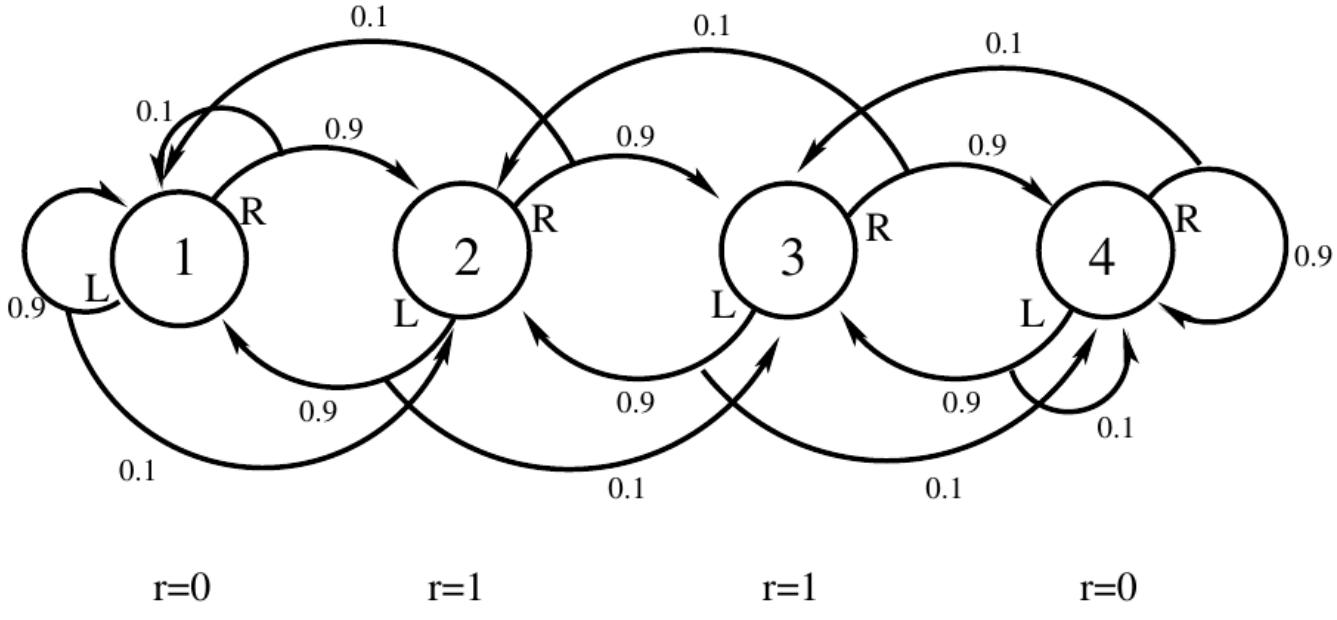
Second Layer: LSPI

- This layer is involved in control problem for planning trajectories and may be tailored to cope with complex dynamic problems
- The used algorithm is Least Square Policy Iteration
  - It's generic since it reduces the control problem to a Markov Decision Process
  - It's adaptable to different environments since it can learn the underlying process

- To explain LSPI it's needed an overview of:
  1. Markov Decision Process
  2. Policy Iteration and Reinforcement Learning

LSPI - Markov Decision Process

- A MDP is composed by: states S; actions A; a probability P to do a transition between states through an action; rewards or costs R for a transition; a discount factor $\gamma \in [0, 1)$ for future rewards
- A deterministic policy $\pi(s)$ indicates the action to take on s



LSPI - Markov Decision Process

- The state-action value function $Q^{\pi}(s, a)$ indicates the expected, discounted, total reward when taking action a in state s and following policy π thereafter
- Maximizing Q^{π} on s and varying $a \in A$ it's possible to find the action that locally maximizes the total reward
  - Policy Iteration permits to discover an optimal decision policy

$$\pi_{m+1}(s) = \arg\max_{a \in A} Q^{\pi_m}(s, a)$$

- Problems
  - Computing Q^{π} on a continuous space requires exponential time and wastes space
  - The state-action function is unknown

LSPI

- The decisional policy is discovered from samples $< s, a, r, s' >$ produced by observations of process or a generative model
- Q is linearly approximated so that it can be efficiently computed and stored
  - Approximation is iteratively refined

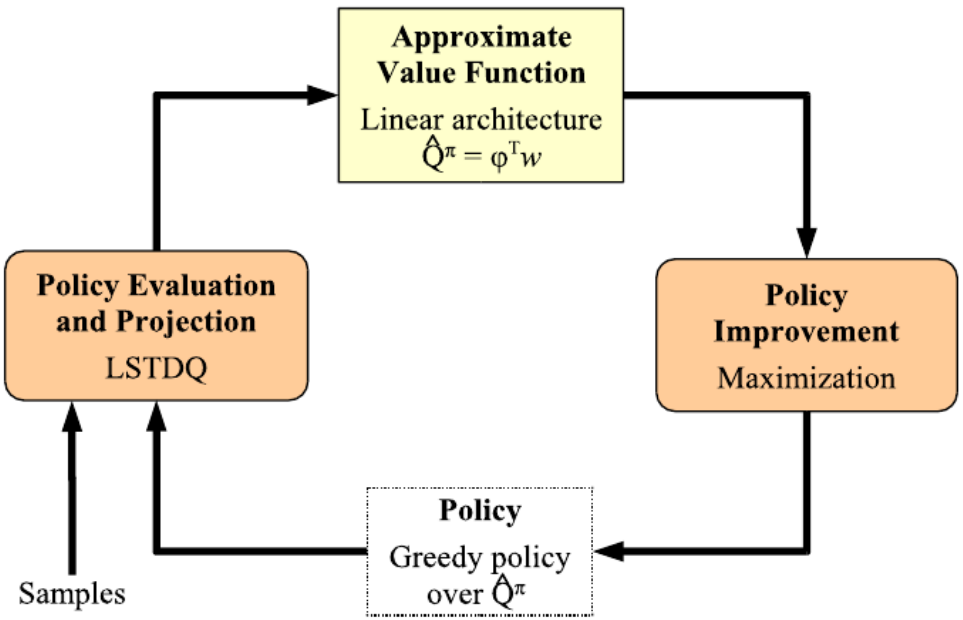
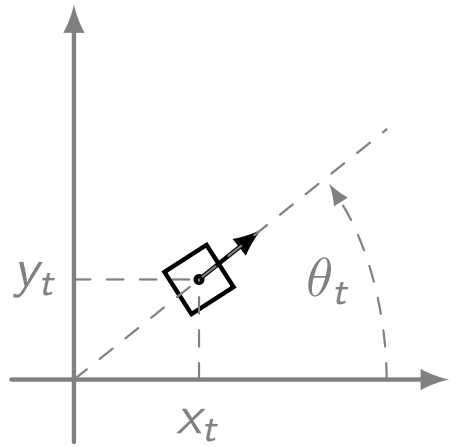


Figure 3: Least-squares policy iteration.

A*-LSPI

- The mobile robot has:
  - An omni-directional wheel and two driving wheels
  - Six ultrasonic sensors in the front part with detection distance d and detection angles 30°

- The state of the robot at time t can be expressed as $[x_t, y_t, \theta_t]$
  - A path in the plane is composed of a finite number of states



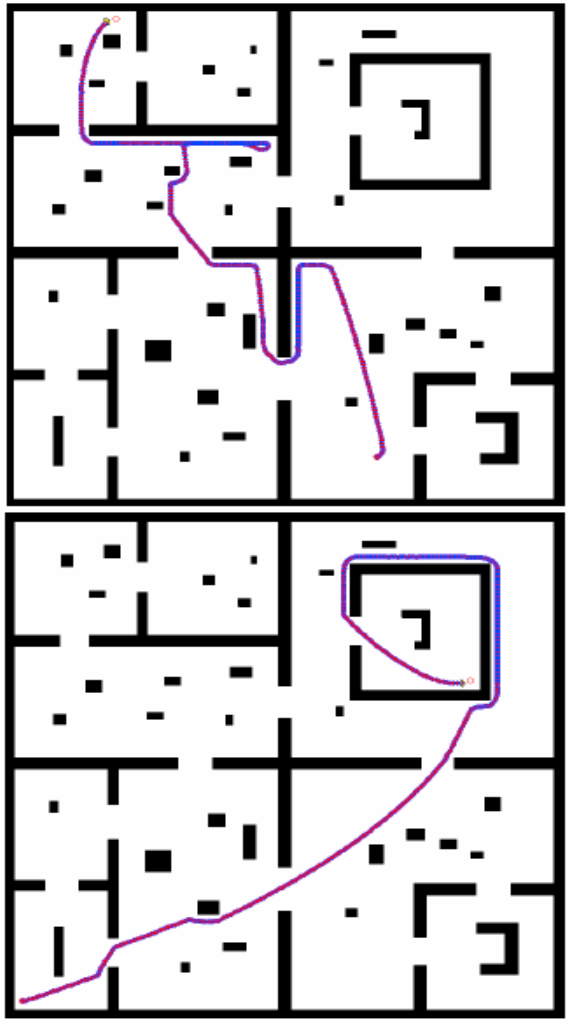
- The problem of path planning can be modelled with two MDP:
  - A MDP is used to approach the target, the other to avoid obstacles

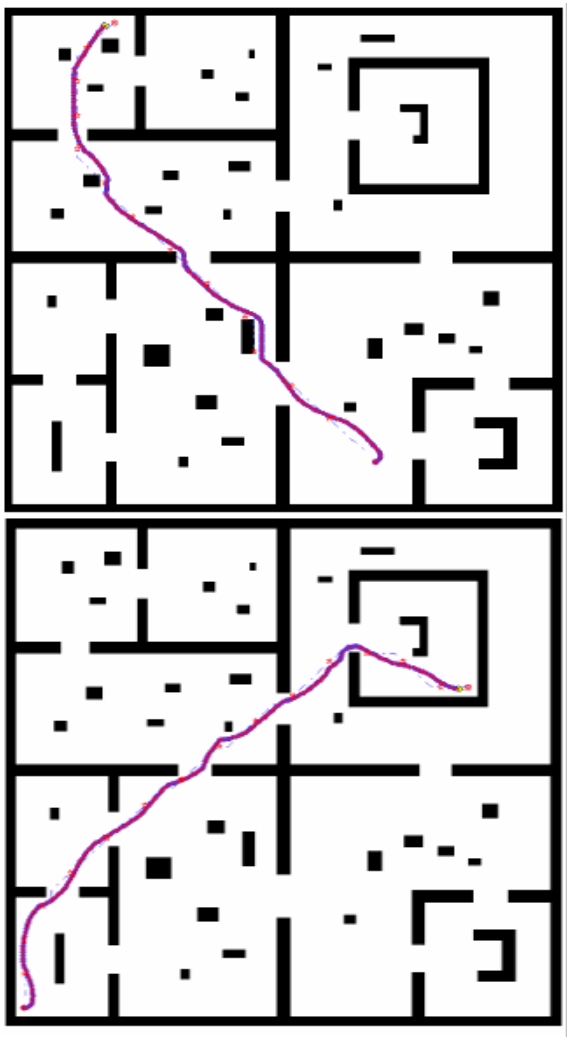
A*-LSPI - MDP Models

"LSPI pushes the robot closer to the target and away from obstacles maximizing the obtained rewards"

	The MDP model of approaching the goal	The MDP model of learning to avoid obstacles
state	$[d_g,\ \phi_g\ ]$	(S1,S2, S3,S4, S5,S6)
action	The combination of the speed of left and right wheel: forward: [0.5, 0.5] turn right: [0, 0.5] turn left: [0.5, 0]	The combination of the speeds of the left and right wheel: forward: [0.5, 0.5] turn right: [0, 0.5] turn left: [0.5, 0]
reward	$\begin{cases} 1\ , \text{ if the robot achieve the goal} \\ -d_g - \phi_g\ , \text{ else} \end{cases}$	See Table 2, where k=0.9

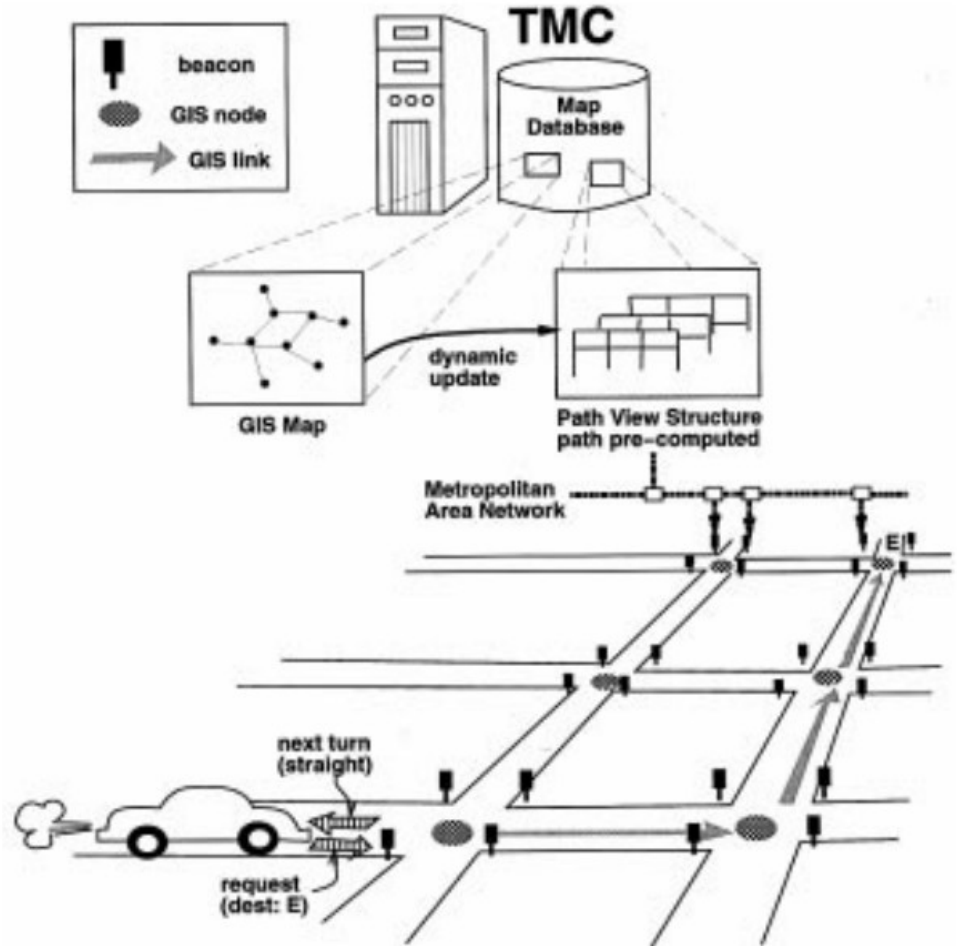
A*-LSPI - Experimental results





Path Finding in ITS

- ITS are systems in which information and communication technologies are applied in the field of road transport
  - Park assistance, traffic control, car navigation, etc...



Path Finding in ITS - Required Features

- Stringent time constraints for path finding
  - Vehicles can walk through beacons
  - Many vehicles are on road during rush hours
  - All pairs shortest path could be pre-computed
- Route reorganization basing on up-to-date traffic data
  - If the paths are pre-computed they must be often updated
- Road networks may have up to 20,000 nodes
  - Common shortest path algorithms are not suitable
    - Dijkstra $O(m + n \log n)$ for node
  - Common memory representations are too expensive
    - Adjacency matrix $O(n^2)$

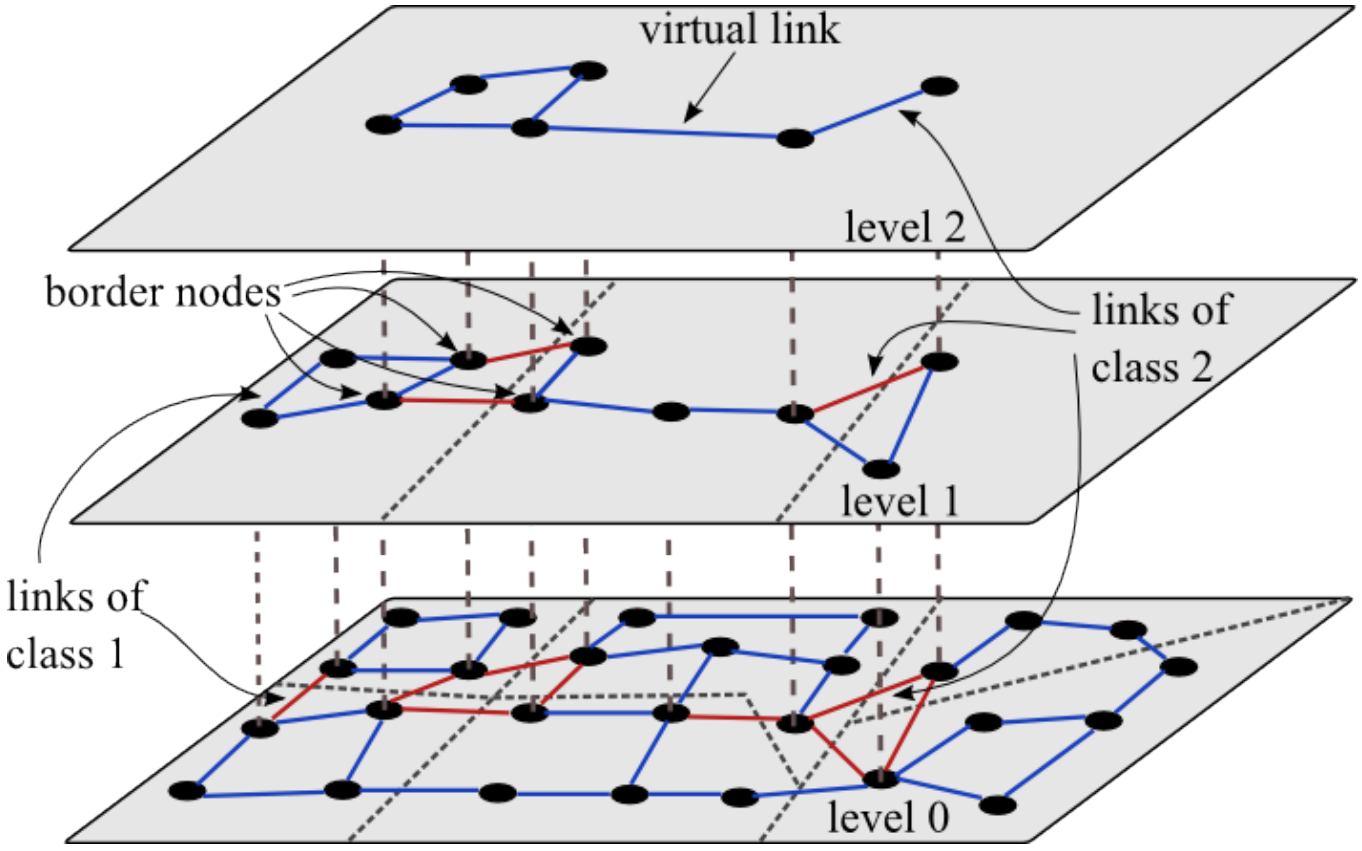
A HPVM for Path Finding in ITS

"It's usual in a travel to go from a local road to the highway and the reverse once the destination's been reached"

- The HPVM solution propose a fragmentation of the graph and a hierarchical organization to solve memory and computation issues
- The solution is organized in:
  - Hierarchy Generation
  - Path Computation and Maintenance
  - Path Finding

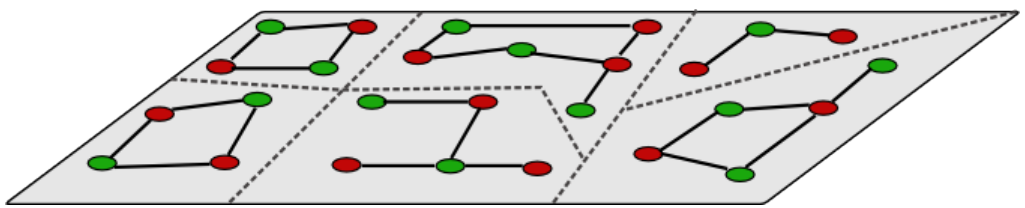
HPVM - Hierarchy Generation

- Links are classified by the maximum speed of travel
  - Classes 0, 1, 2
- Hierarchy is generated iteratively after fragmentation



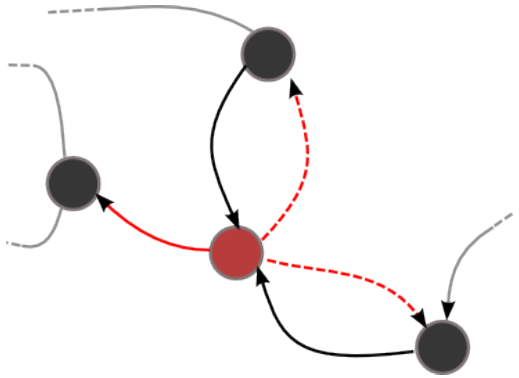
HPVM - Path Generation

- Shortest path are computed for each region at each level
- Two-Color Dijkstra is a speed-up of Dijkstra algorithm based on a two colouring technique:
  - Each node is red or green, green nodes have only red neighbours
  - Compute Dijkstra for red nodes and extend to greens
    $O(r \times n \log n + g \times n)$
- Memory requirements are reduced thanks to fragmentation

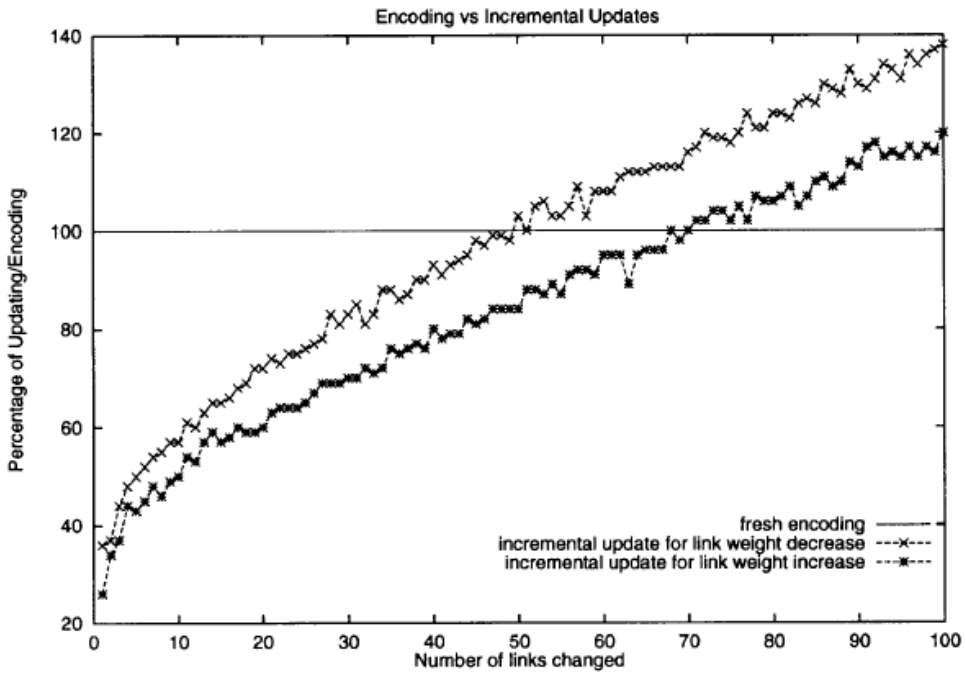


HPVM - Path Maintenance

- Since update interval must be short a small number of links will change in that time
- It's possible to recompute shortest paths around the changed link and eventually propagate back the changes



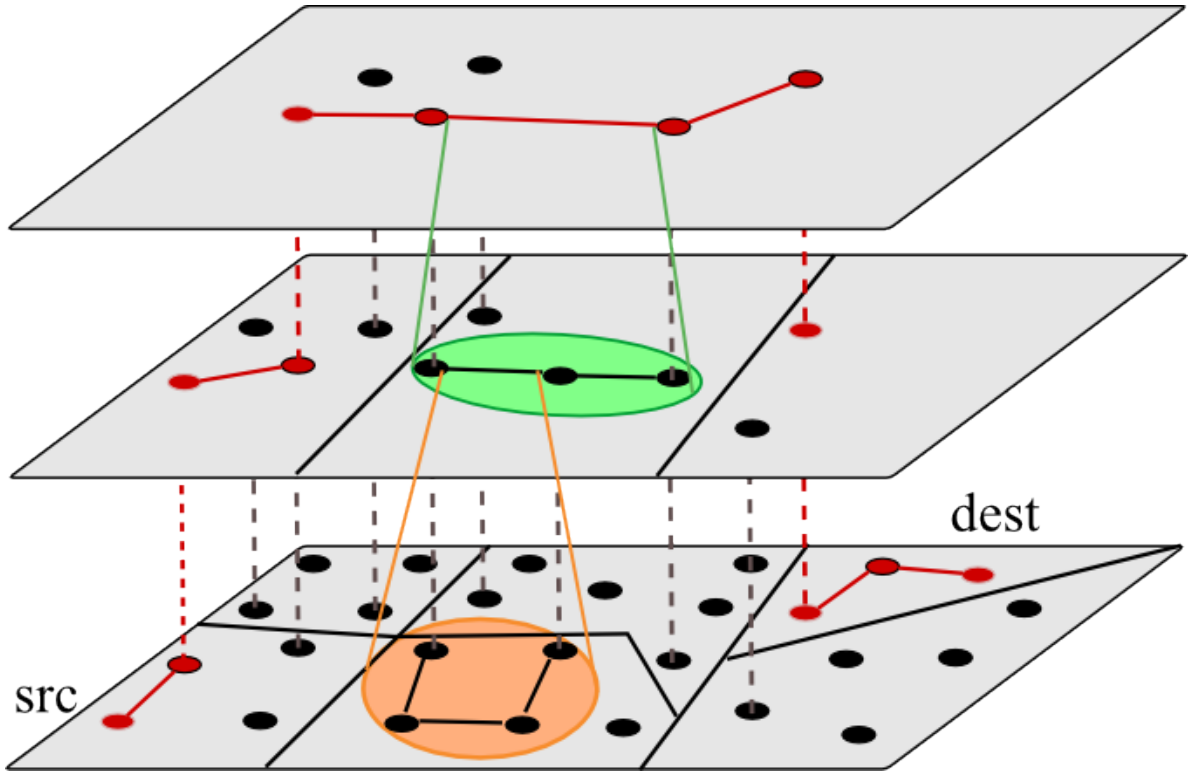
- If many links weight change the update is executed more than once on a node
- When many links change it's useful to recompute paths



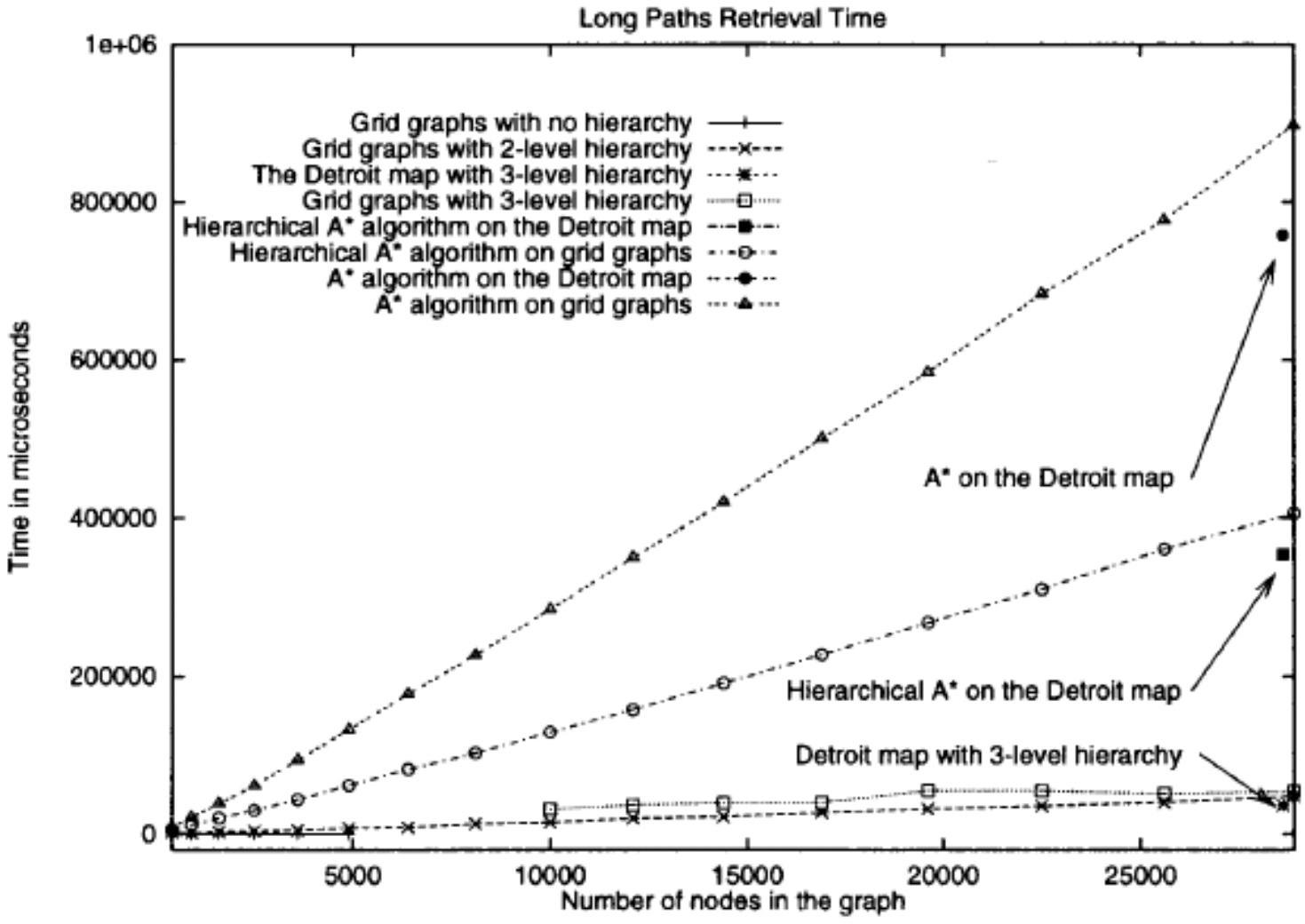
HPVM - Path Finding

"Hierarchy is a less granular view of the network, based on road classes."

- It extimates the best entry/exit point for each region upward/downward in the hierarchy



HPVM - Experimental Results



Conclusions

- Shortest Path Finding is present in many practical situations
  - Algorithms are not optimal
  - They usually balance performance and optimality
- Heuristic
  - Algorithms run faster
  - Paths are near to optimal
- Hierarchy
  - It permits to make a better use of computation resources
  - It can be used to avoid different tasks as in A*-LSPI

The end

Thanks for your attention :)