RUTGERS
School of Arts and Sciences

PRACSYS
Physics-aware Research on Autonomous Computational Systems

Progress in Algorithmic Motion Planning Related to Cloud Robotics, Automation and Manufacturing

Kostas Bekris
Department of Computer Science
Rutgers University

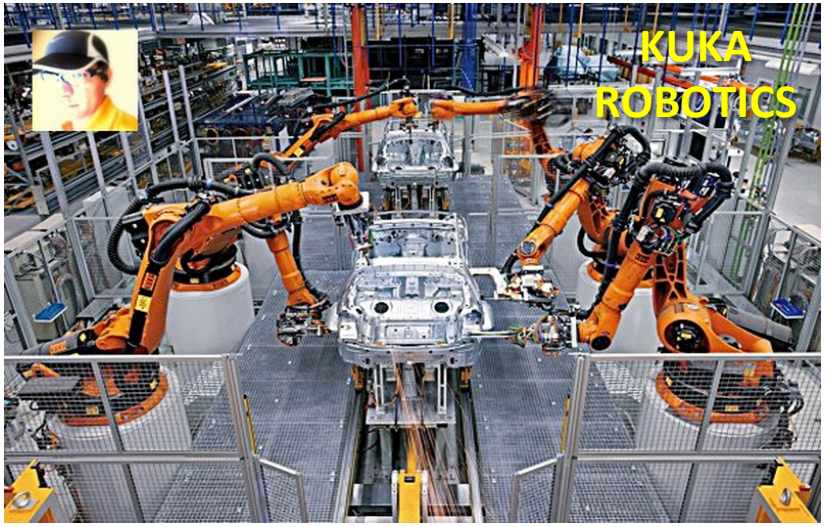
August 17, 2013
Cloud Manufacturing workshop @ IEEE CASE

How is algorithmic development for robots influenced by the availability of a cloud computing infrastructure?

Highlight work in two areas:

- Motion Planning with Popular Sampling-based Algorithms
- Multi-Robot Path Planning on Graph-based Representations

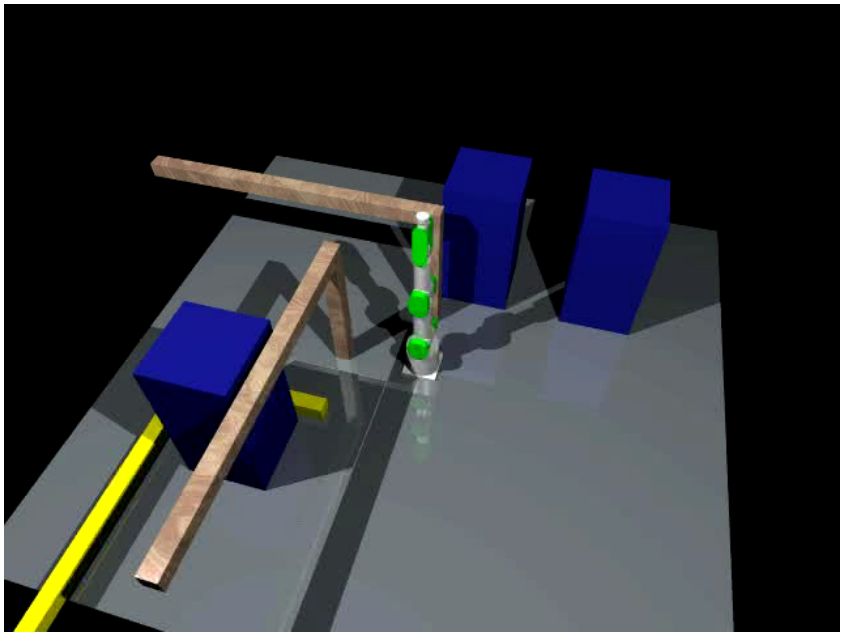
Related Applications



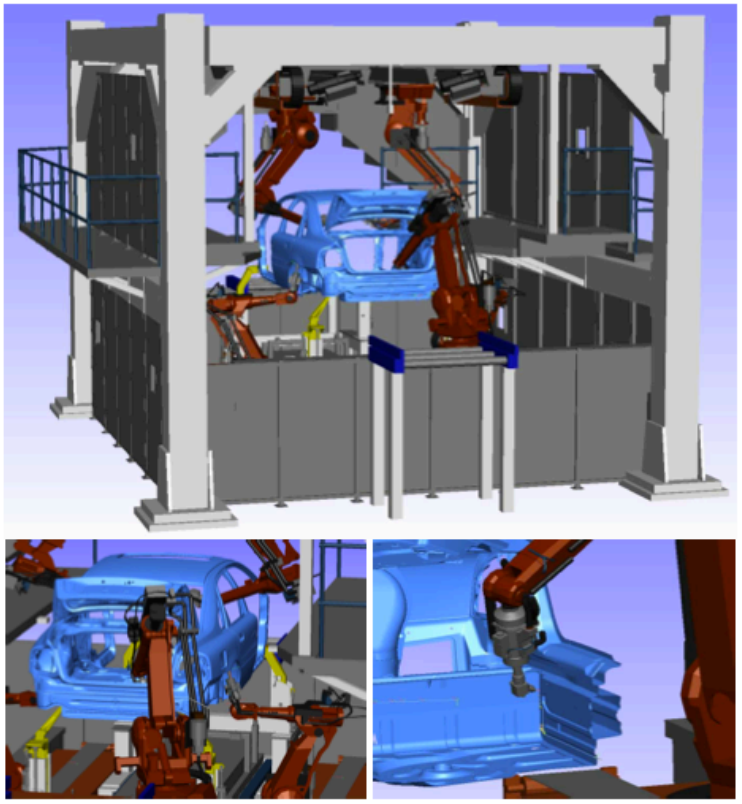
Flexible Manufacturing

Adaptive Distribution Centers

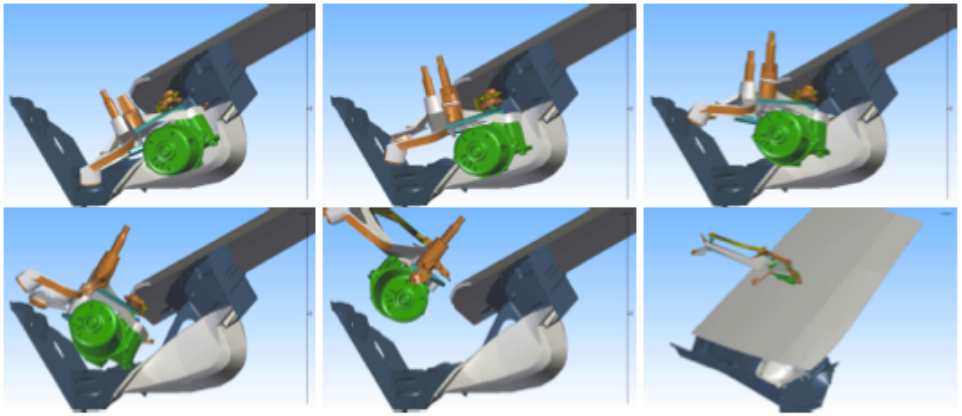
1. Motion Planning



Kavraki Lab, Rice University



Application of motion planning in the manufacturing process of Volvo cars

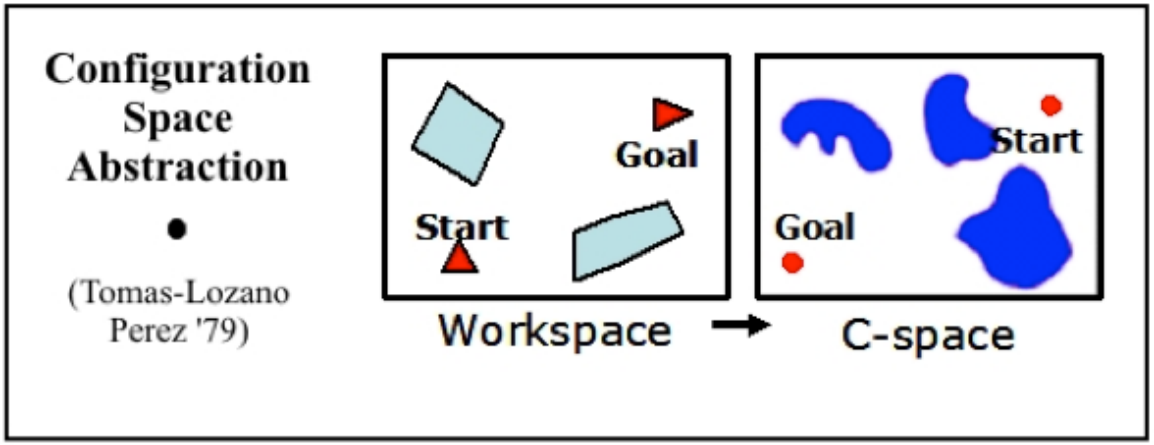


Wiper removal from car body cavity (Kineo CAM)

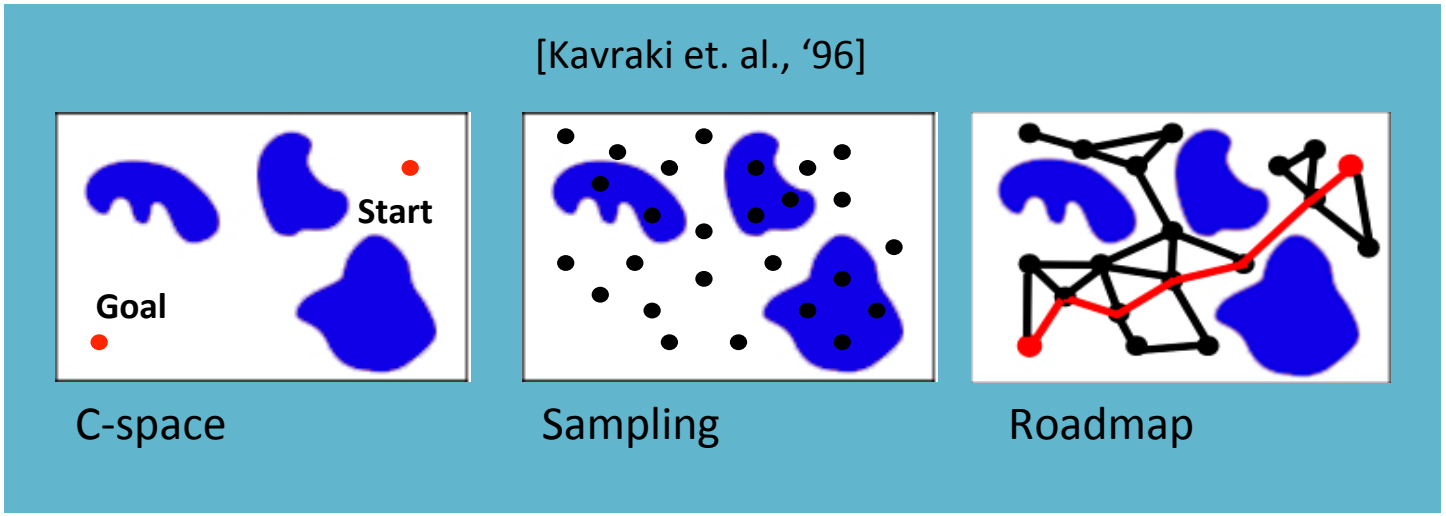
Examples of Industry Adopters:
BMW, Airbus, Ford, GE,
Optivus, Renault,
UGS Technomatix

Sampling-based Motion Planning

- Helpful abstraction:



- Popular solution: Build a data-structure for path planning in the configuration space using sampling to deal with complexity



How should such motion planning algorithms operate in new manufacturing environments that utilize cloud computing?

Setup

Data Forwarding

Data Forwarding

Command Forwarding

Moving in and altering the workspace

Data Collection

Data Collection

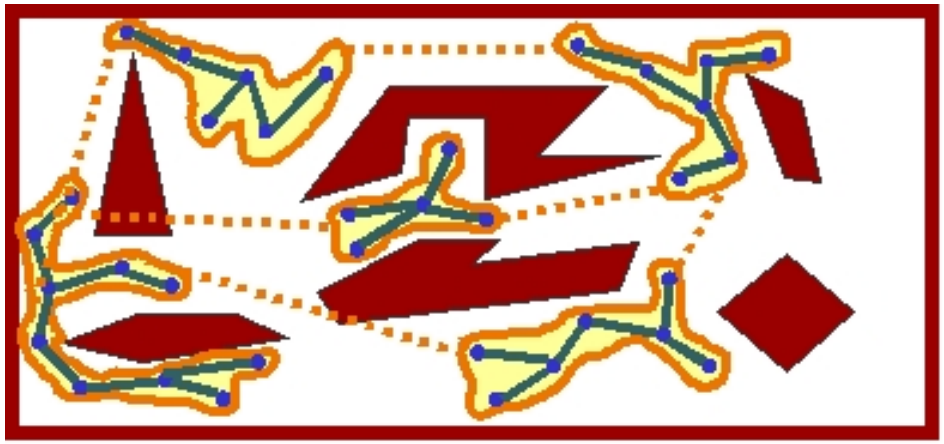
Cloud Provides Parallelization

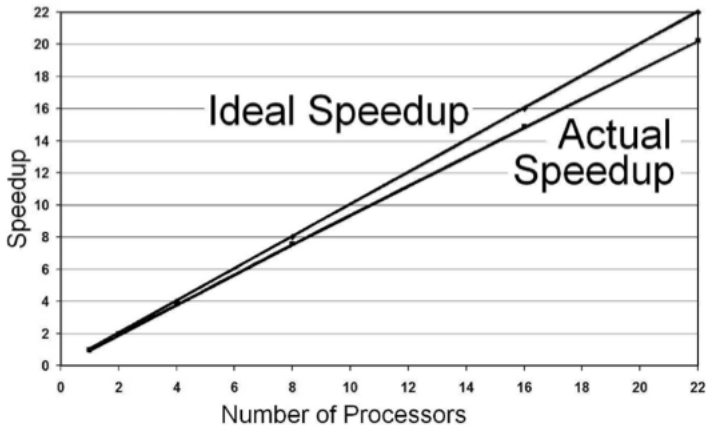
Sampling-based algorithms are highly parallelizable

(even called "embarassingly parallel" [Amato et al. ICRA '99])

- Computation of collision-free nodes can be achieved independently
- Identification of edges introduces dependency between processors

Appropriate distributed solutions exist (Sampling-based Roadmap of Trees method [Bekris et al. IROS '04, Plaku et al. IEEE TRO '05])

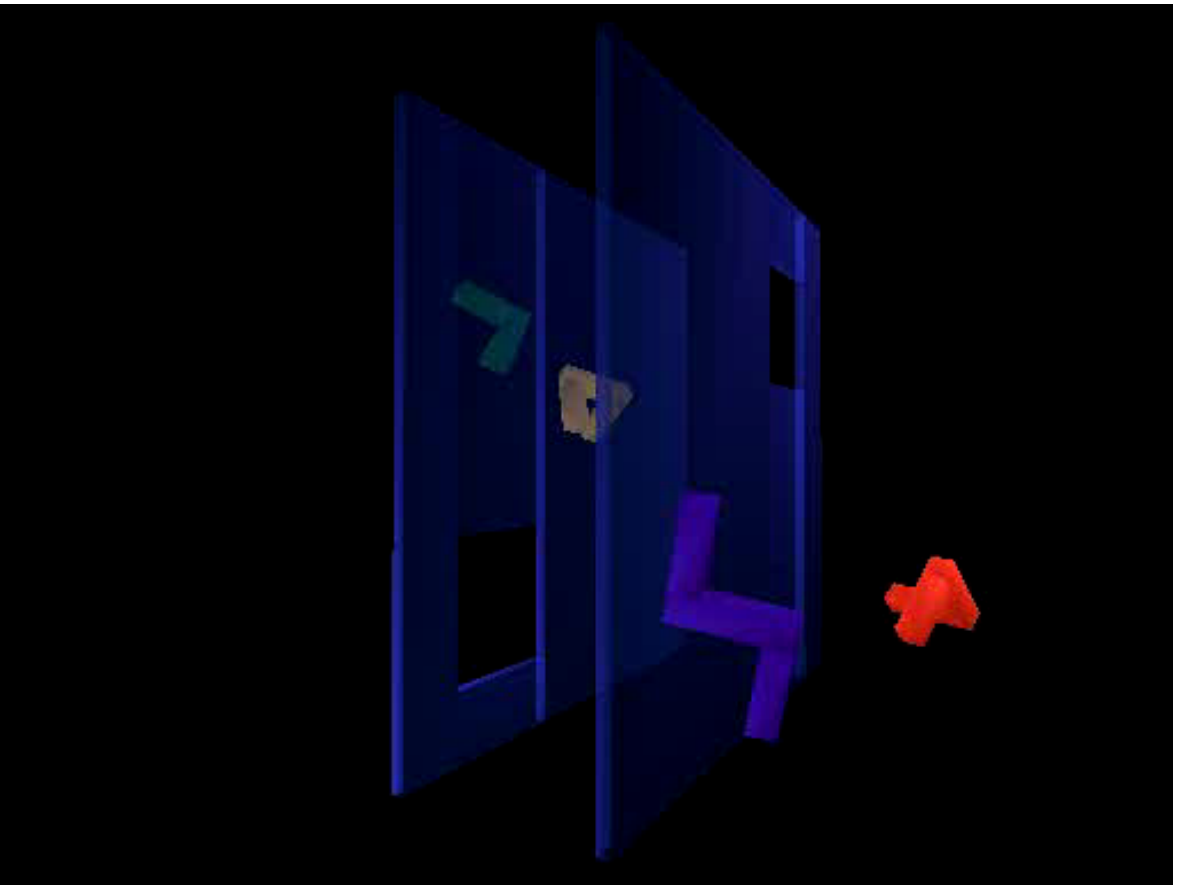




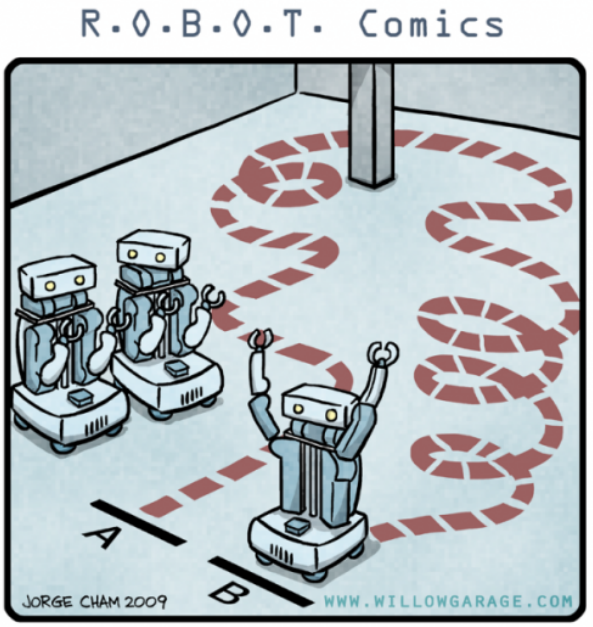
Appropriate distributed versions of RRT were also studied recently [Devaurs, Siméon, Cortés TRO '13]

Path Quality

Early solutions focused on feasibility and computational efficiency, sacrificing path quality



Solution path after smoothing from
Sampling-based Roadmap of Trees (SRT) method
[Bekris et al. IROS '04, Plaku et al. IEEE TRO '05]

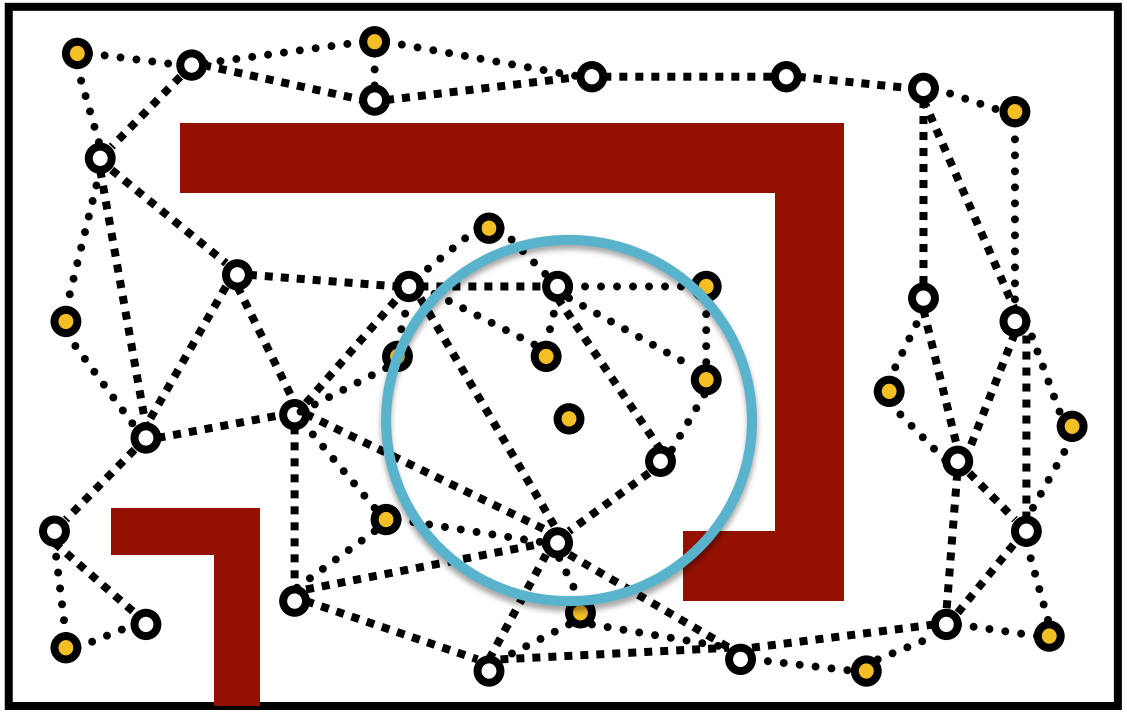


"HIS PATH-PLANNING MAY BE SUB-OPTIMAL, BUT IT'S GOT FLAIR."

Path Quality Breakthrough

When does a roadmap return optimal paths?

- A fully connected graph in the state space will give an asymptotically optimal solution
- Computationally infeasible (i.e., resembles exhaustive search)



From percolation theory

It is sufficient if we attempt to connect any new sample with approximately *logn* neighbors, where *n* is the number of nodes in the roadmap.

[PRM* - Karaman, Frazzoli '11]

Resulting data structures are still large/dense but cloud computing makes their computation easier

Appropriate Level of Abstraction

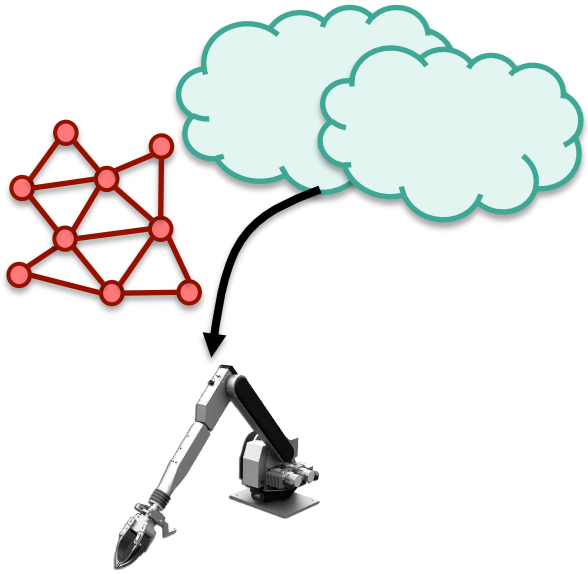
Which aspects of path planning should take place on the cloud and which should take place on the robot?

Many choices available – appropriate abstraction will depend on the application:

- e.g., compute paths directly on the cloud and transmit them

Alternative:

- The cloud updates a roadmap given global sensory data, which is periodically transmitted to the robot
- The robot computes paths on the roadmap considering local sensory input, e.g., avoiding local moving obstacles



New Requirements

Such mode of operation introduces new requirements:

- We need small-size, sparse data structures that still provide good quality solutions

Small-size, sparse roadmaps allow for:

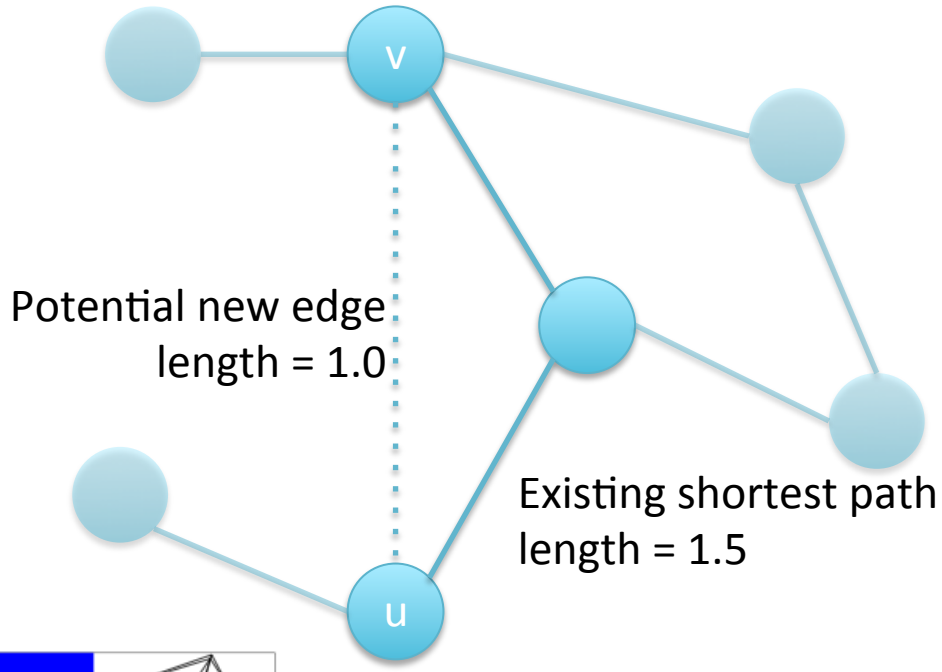
- Efficient, fast communication over a wireless infrastructure
- Easy storage on a resource constrained robot
- Fast updates given local sensory information on the actual robot
- Fast query resolution given dynamically generating queries

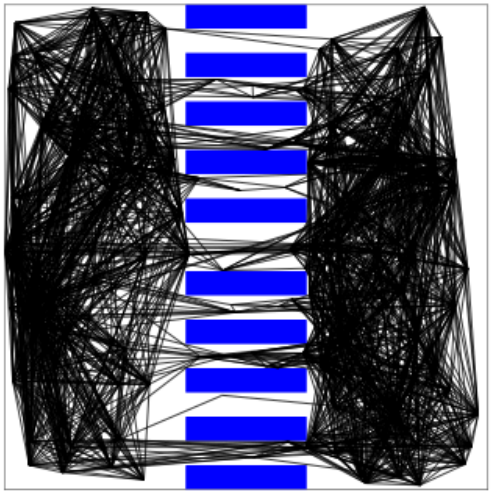
Satisfy the theoretical objective of such data structures:

- Compact representations which are quick to query.
- Representations which truly reflect the connectivity of the C-space., i.e., continuous space spanners. [Agarwal, IROS workshop '11]

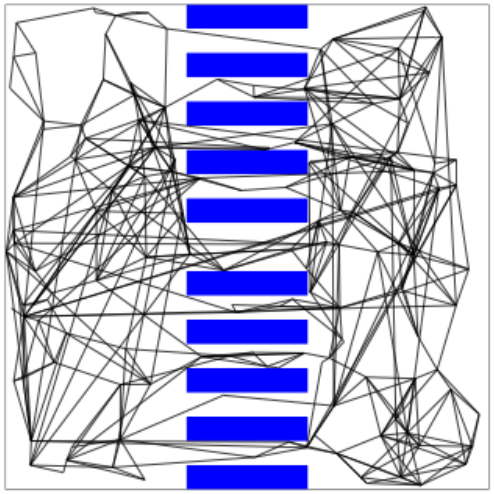
Asymptotic Near-Optimality

- A *t*-spanner is a sparse subgraph
- For every shortest path in the original graph
  - There is a path in the spanner that is no longer than *t* times the original length





(a) Roadmap with 2346 edges



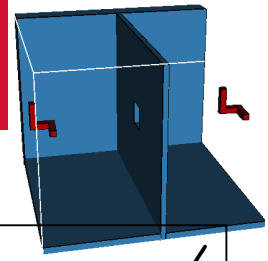
(b) Spanner with 470 edges

Giving rise to a sequential approach:

- Compute k-PRM*
- Return its spanner

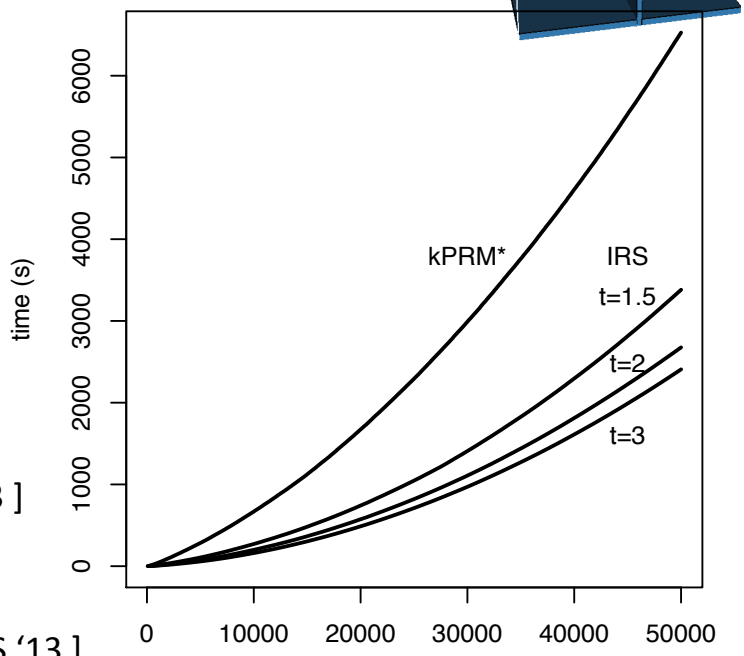
[Marble, Bekris IROS '11]
[Based on the graph spanner approach by Baswana, Sen '07]

Incremental Roadmap Spanner

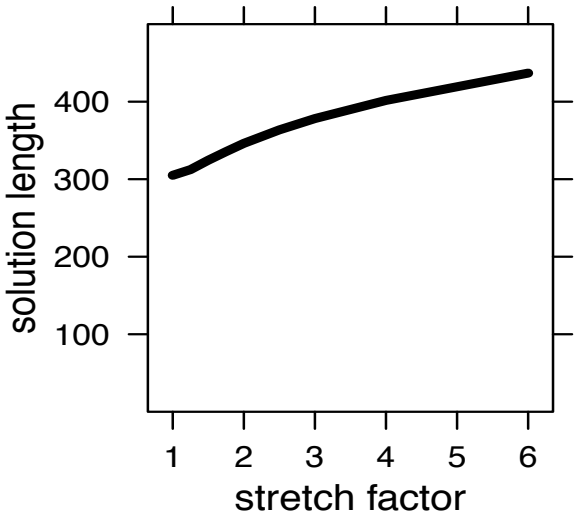


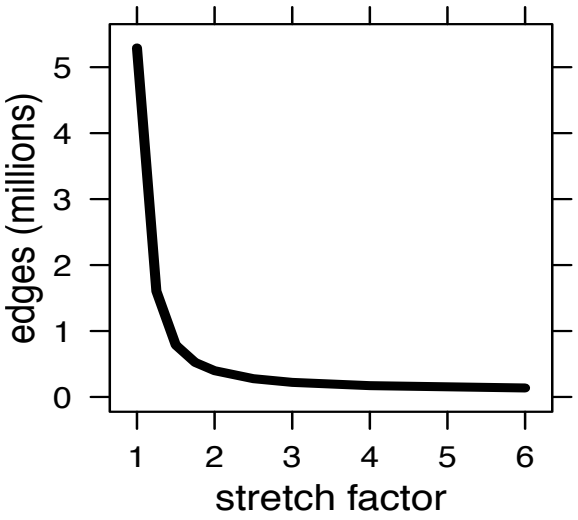
- Start with the asymptotically optimal k-PRM*
- Interleave an incremental spanner algorithm
- Result: An asymptotically *near*-optimal planner
  - Smaller average increase in path length than the stretch factor
  - Sparse roadmap with smaller memory footprint
  - Faster construction and online query resolution

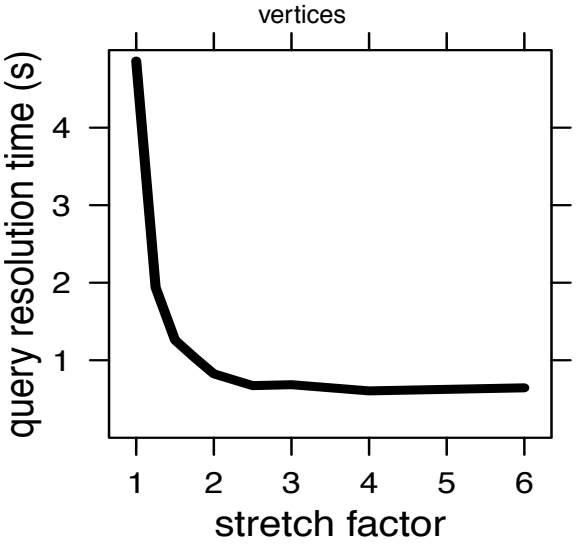
[Marble, Bekris TRO '13]



- Alternative methods with same objectives recently proposed [Saltzman, Halperin et al. ICRA '13 - Wang, Balkcom IROS '13]





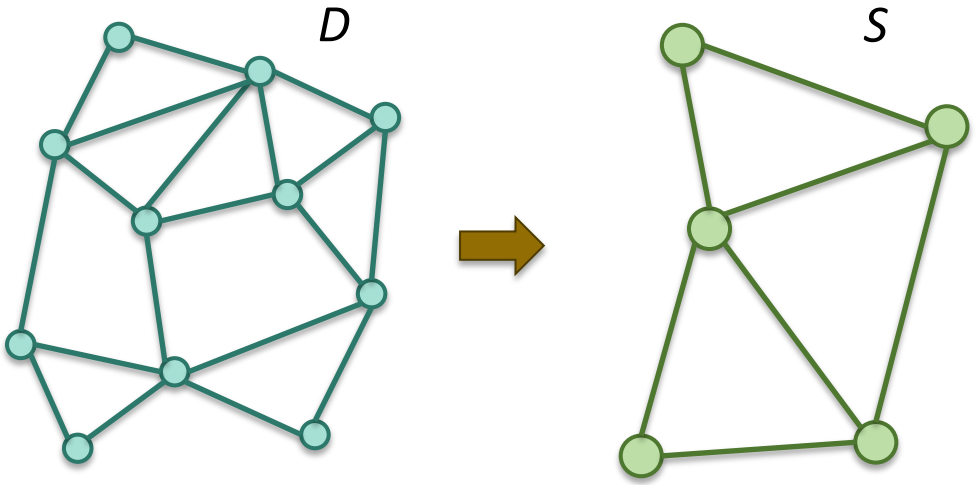


Sparse Roadmap Spanner (SPARS)

- Spanners maintain all the nodes of the original graph
- In continuous spaces, not all nodes are necessary for near-optimality

[Dobson, Krontiris, Bekris WAFR '12, IJRR '13 (to appear)]

- Consider two graphs in parallel:



Dense Graph:

– Asymptotically Optimal (δ-PRM*)

Sparse Roadmap:

– Asympt. Near-Optimal
– A small subset of the samples is selected as nodes

- When should samples be added to *S*?
  – If necessary for coverage, connectivity, optimality

- Available through the ROS-supported Open-Motion Planning Library (OMPL)

Caveat: Not as easily parallelizable

Finite-time Properties

- All existing guarantees are asymptotic in nature

- Looking into properties that can be achieved after finite computation time:

Probabilistic Near-Optimality in Finite Time

✓ Algorithm provides a confidence probability *p* of returning a path after iterations n, which:

  - will be near-optimal, i.e., $|\pi_{PNO}| \leq a\ |\pi^*| + b$ for real-valued *a* and *b*

[Dobson, Bekris IROS '13]

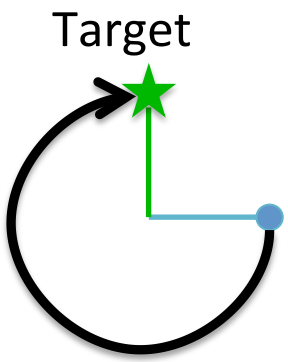
- Current solution computationally more expensive than PRM*
  - Requires significant computational resources to be achieved
  - But it is easily parallelizable

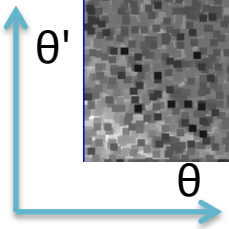
Many Interesting Directions

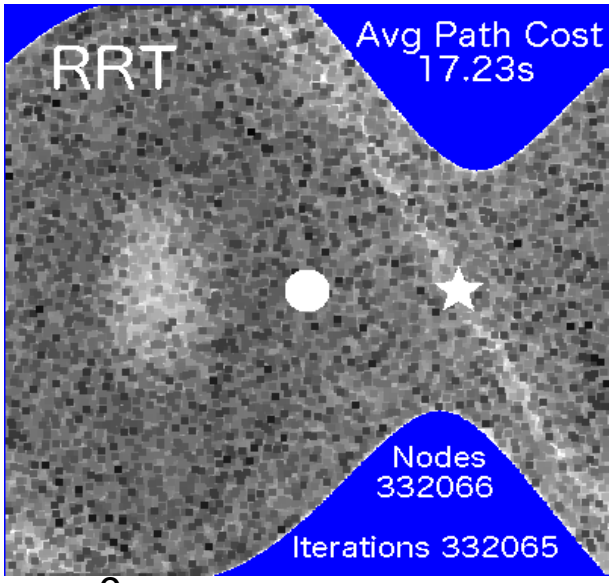
- Motion planning algorithms with finite-time properties are appropriate for integration with task planners
  - How such integrations can appropriately utilize cloud computing?
- Challenges that involve dynamics, physics-based simulation and planning under uncertainty [Littlefield, Bekris IROS '13]
  - Physics-based simulation is a powerful tool that is computationally expensive

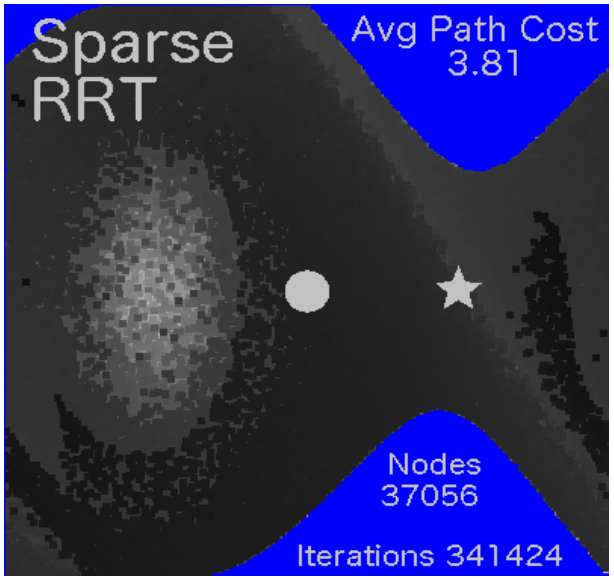
Visualization of the best path cost to each state in the phase space of a pendulum.

Darker colors correspond to better quality paths.









Algorithms used same computation time

2. Larger-scale Path Planning

Some motivating path planning challenges involve thousands of moving/movable objects:

- Adaptive distribution centers
- Container handling at ports

More of a scalability challenge than a kinematics/dynamics issue

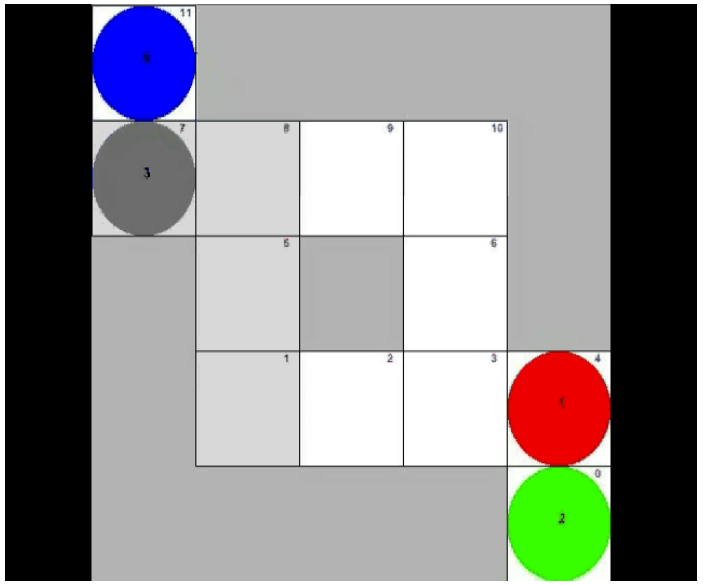
Cloud computing is already applied in this area

- Mostly heuristic solutions in nature

Does the availability of cloud computing allow the achievement of certain guarantees fast enough?

Multi-agent Pathfinding on a Graph

How can agents move on a graph from an initial assignment to a goal assignment without two of them occupying the same node simultaneously?



It doesn't include every aspect of the motivating applications (e.g., task assignment, dynamic goal generation)

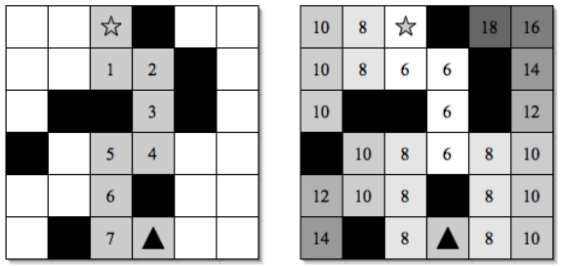
- It is the core path planning challenge for this type of problems.

Many variations of this basic challenge can be defined and have been studied

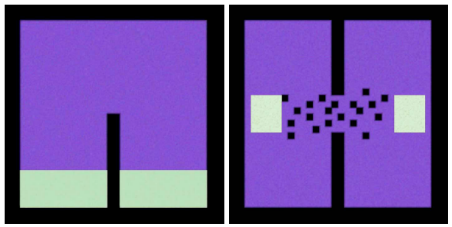
e.g., many agents can share goals, agents can move sequentially or in parallel, etc

Incomplete Methods

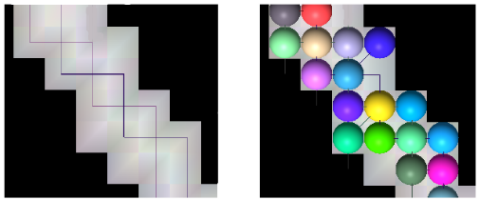
- Computationally efficient.
- Decoupled framework.
- No guarantees for
  - Completeness.
  - Path Quality.



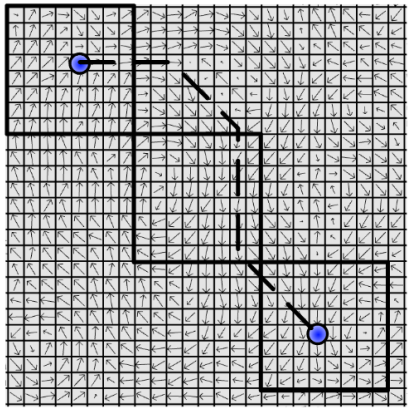
[Silver 2005]



[Sturtevant and Buro 2006]



[Wang and Botea 2008]

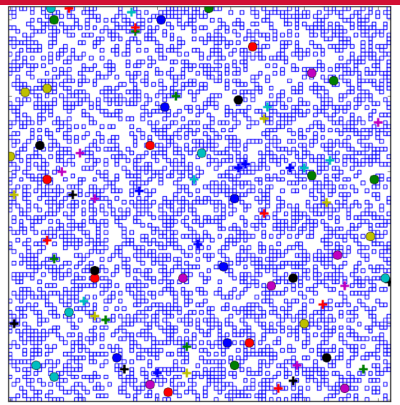


[Jansen and Sturtevant 2008]

- Dynamic prioritization and windowed search [Silver 2005].
- Spatial abstraction with heuristic computation [Sturtevant and Buro 2006].
- Use of a flow network with replanning [Wang and Botea 2008].
- Smart direction maps that learns movements [Jansen and Sturtevant 2008].

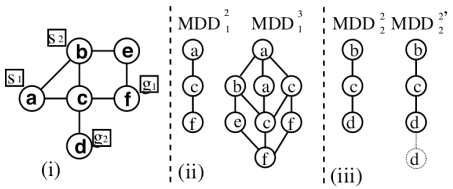
Optimal Methods

- Provide path quality guarantees.
- Coupled framework. Often A*-based.
- Great recent progress but…
  - Scalability shown only up to about 50-60 agents.



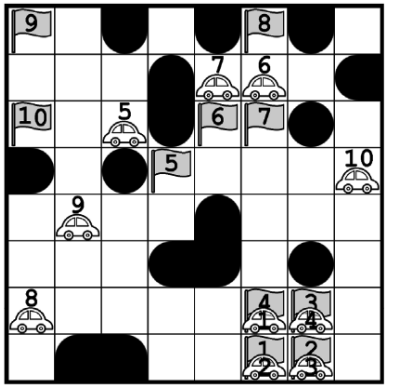
[Wagner and Choset 2013]

- Iterative deepening manner
  [Sharon et al. 2011].



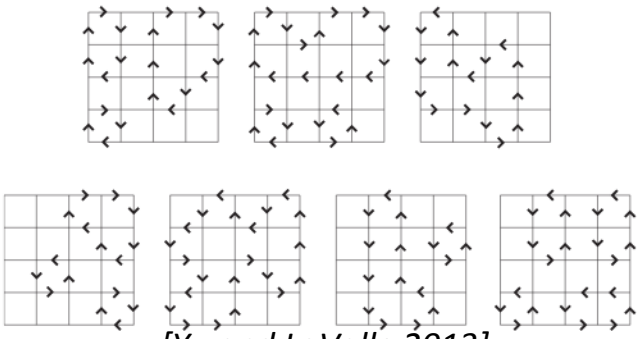
[Sharon et al. 2011]

- Working on independent subproblems
  [Standley 2010, Standley and Korf 2011].



[Standley 2010, Standley and Korf 2011]

- Based on linear programming (ILP)
  [Yu and LaValle 2013].

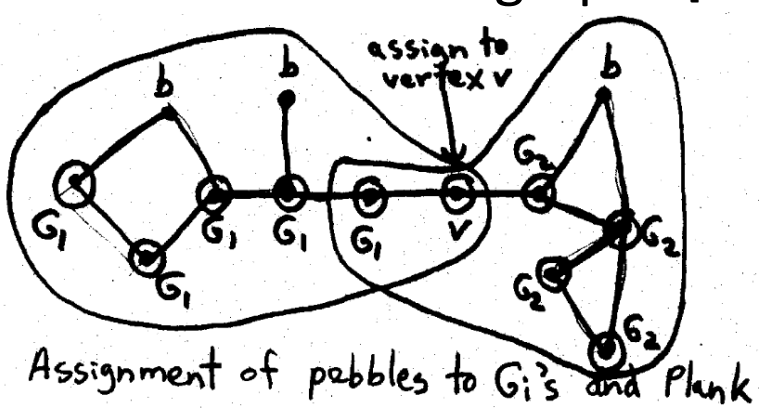


[Yu and LaValle 2013]

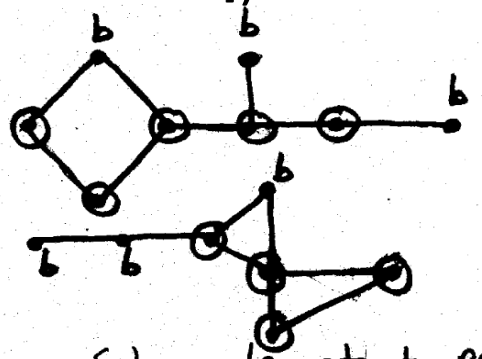
- Subdimensional expansion search space
  [Wagner and Choset 2011].

Checking for Feasibility is (Really) Easy!

- Polynomial time feasibility test algorithms exist for quite some time ("pebble motion on a graph" [Kornhauser et al. 1984])

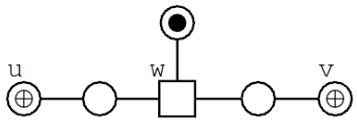


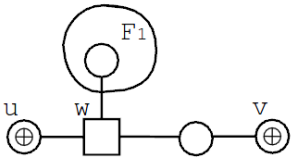
Assignment of pebbles to G_i's and Plank

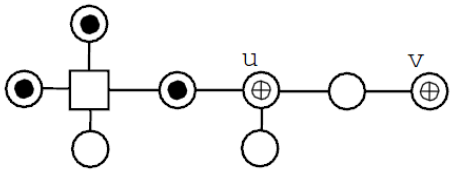


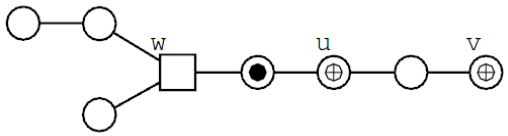
Subpuzzle start positions for pebbling in 3-17(a)

- Linear running time algorithm for trees proposed [Auletta et al. 1999].

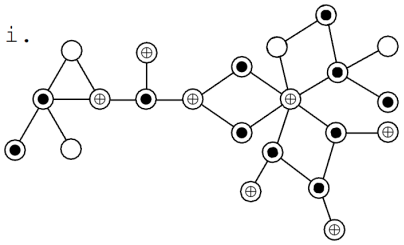


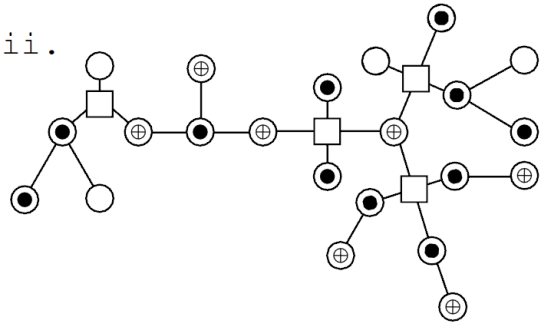






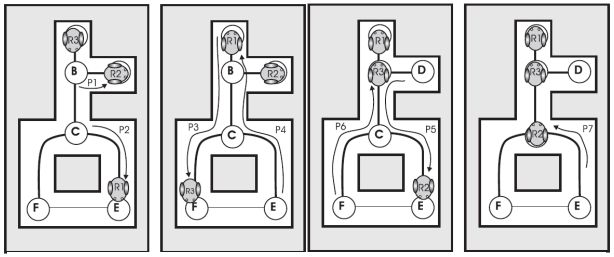
- Linear running time algorithm for graphs with two holes was also recently proposed [Goraly and Hassin 2010].



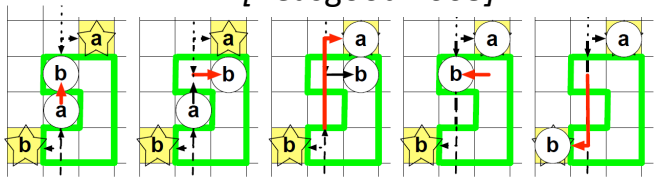


Suboptimal but Complete Methods

- Efficient: polynomial running time.
- They will find a solution if one exists.
- They do not provide optimal paths.

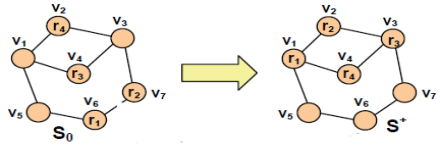


[Peasgood *2008*]

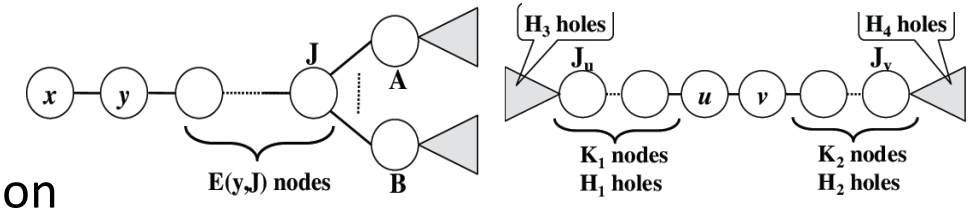


[Wang and Botea *2011*]

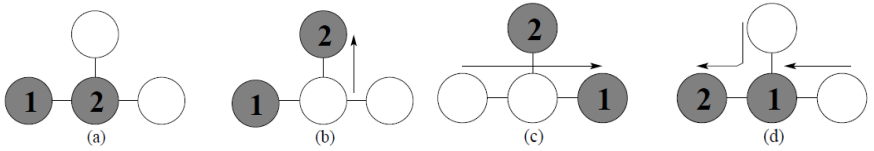
- Specific topological graphs
  [Peasgood et al. 2008].
- Bi-connected graphs with two empty vertices
  [Surynek 2009].
- Slideable grid-based problems
  [Wang and Botea 2011].
- Complete on trees
  [Khorshid et al. 2011].
- Polynomial-time solution (Push&Swap) on graphs with two empty vertices
  [Luna and Bekris IJCAI 2011, Sajid et al. SOCS 2012]
- Incorporate primitives from feasibility tests to improve efficiency
  [Krontiris, Luna, Bekris SoCS 2013]



[Surynek *2009*]



[Khorshid et al. *2011*]

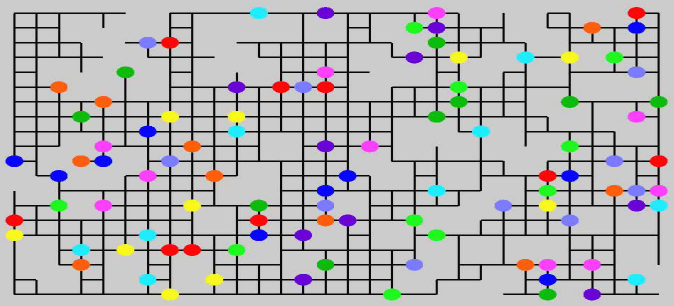


[Luna and Bekris *2011*]

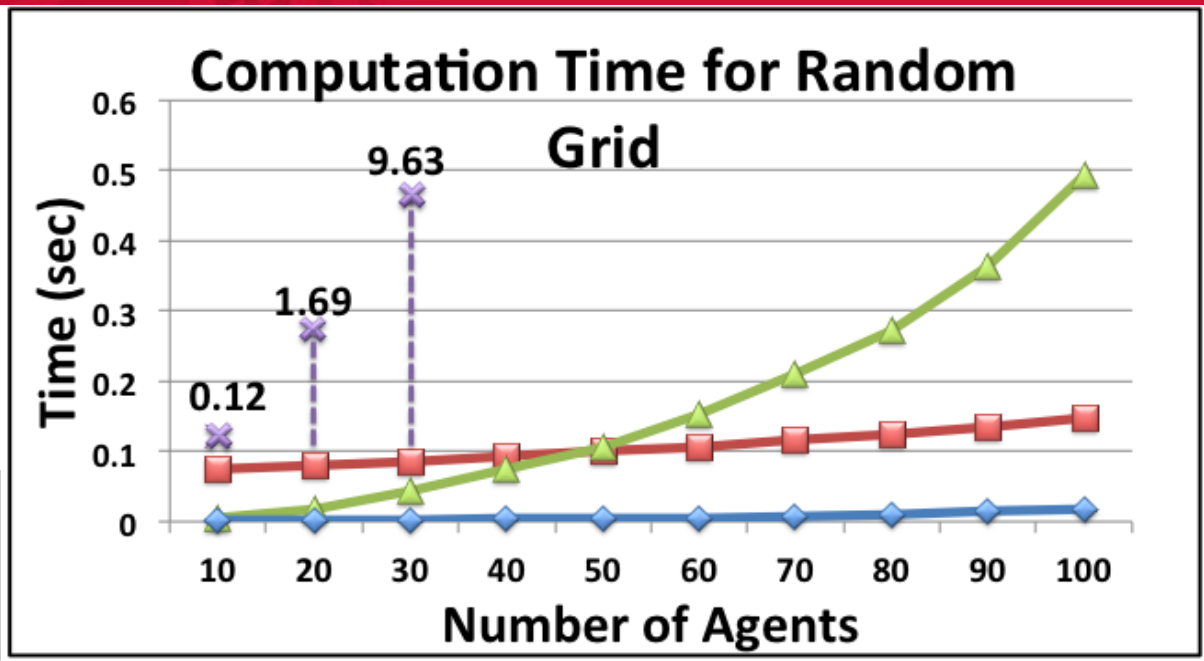
Large Scale Random Grid

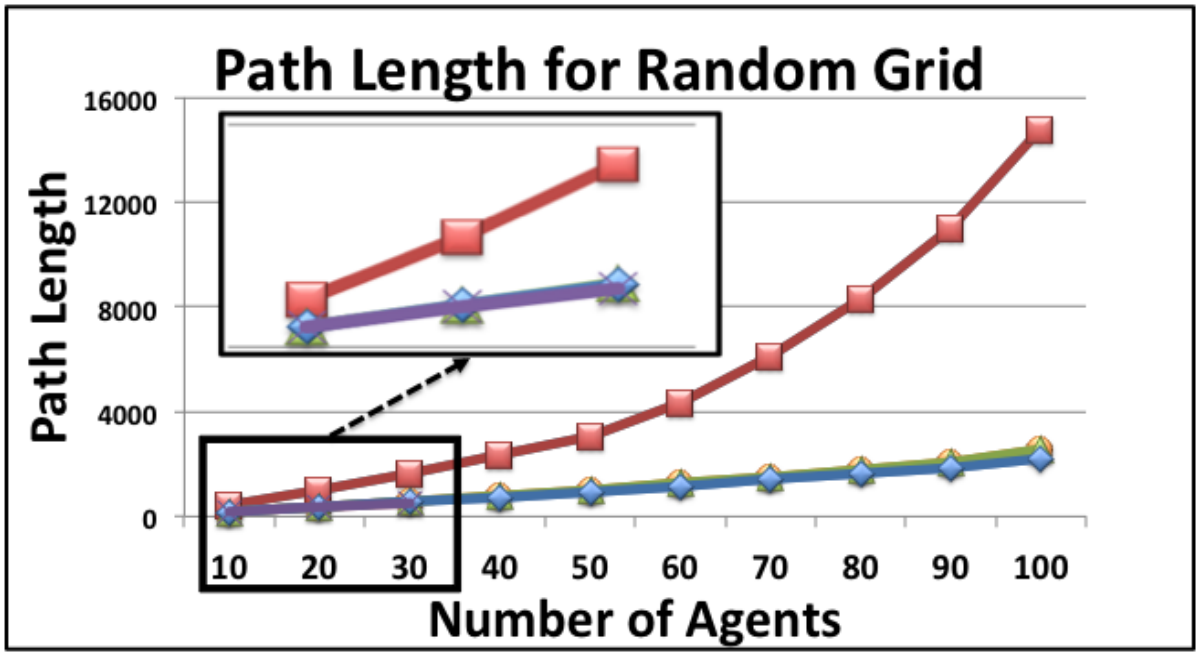
- Random grid: 500 vertices.
- 20% random obstacles.
- From 10 to 100 pebbles.
- 20 runs
- 5 minutes time limit

[Krontiris, Luna, Bekris SoCS 2013]



:ODA*+ID
:Push and Swap
:Feasibility-based
:PMG_Solver

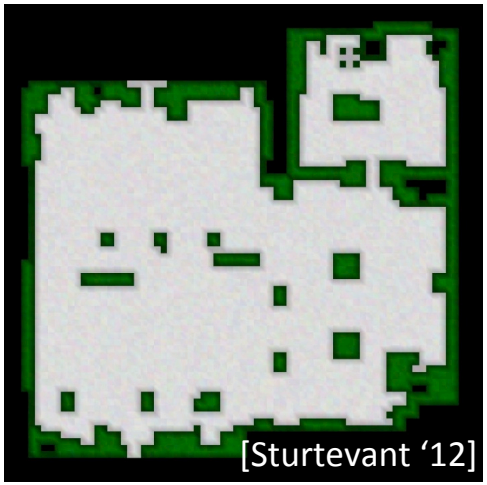




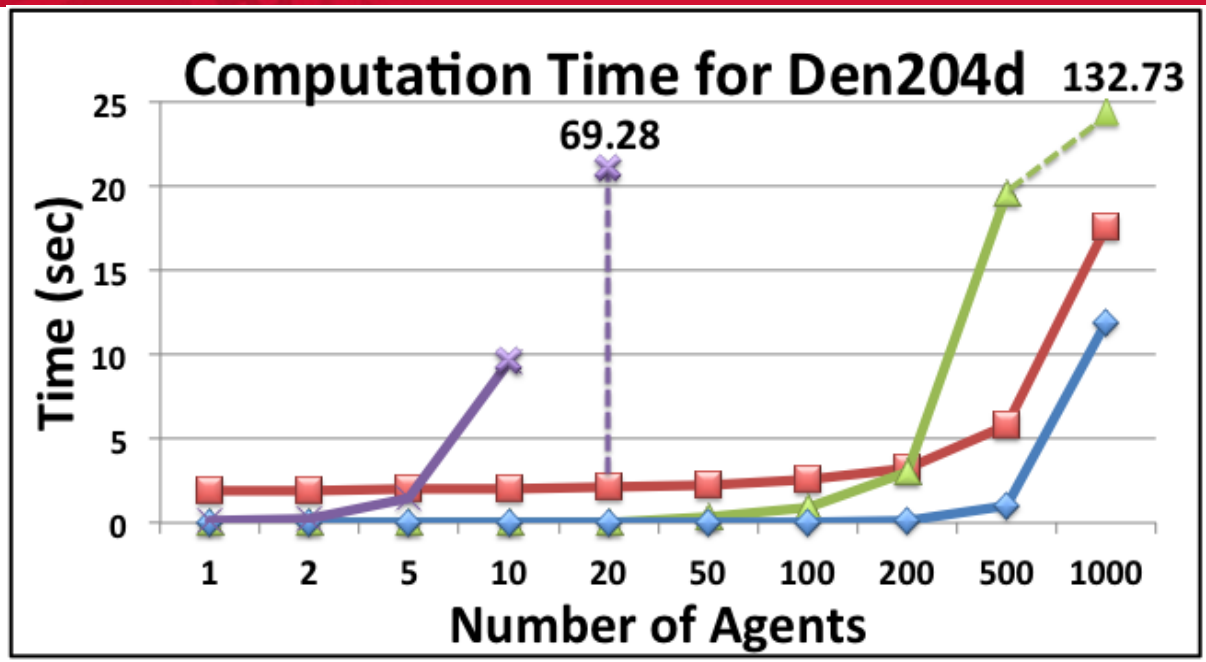
Large Scale Game-based Environment

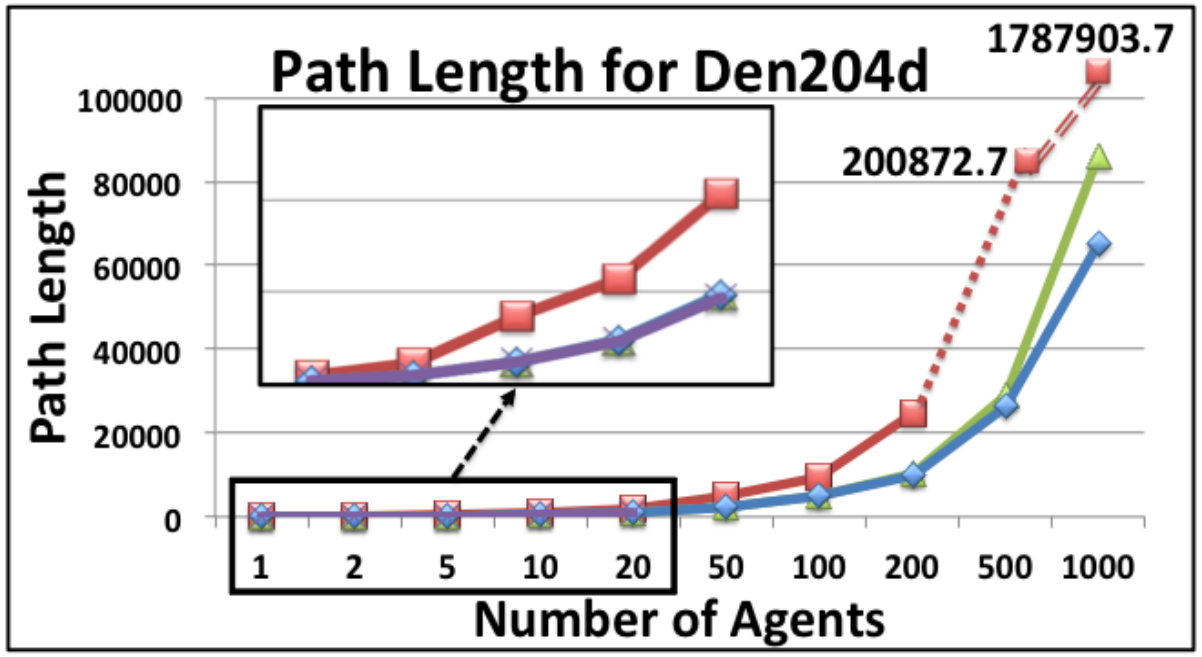
- Game-based world with 2534 vertices.
- From 1 to 1000 pebbles.
- 20 runs
- 5 minutes time limit

[Krontiris, Luna, Bekris SoCS 2013]



:ODA*+ID
:Push and Swap
:Feasibility-based
:PMG_Solver





Discussion

Cloud computing allows us to employ algorithms that provide stronger guarantees.

Key questions:

- What is the appropriate type of guarantees that we should aim for in the era of cloud computing?
- What are additional constraints that we should be respecting, e.g., robustness to communication failures, use of bandwidth and memory requirements?
- What should be locally computed and what should be outsourced to the cloud?

PRACSYS Research Group

Thank you!

http://www.pracsyslab.org

Our research efforts have been supported by:

- the CPS program of the National Science Foundation (NSF),
- the National Aeronautics and Space Administration (NASA),
- internal funds of Rutgers University and the University of Nevada, Reno