POLITECNICO MILANO 1863

IROS MACAU 2019

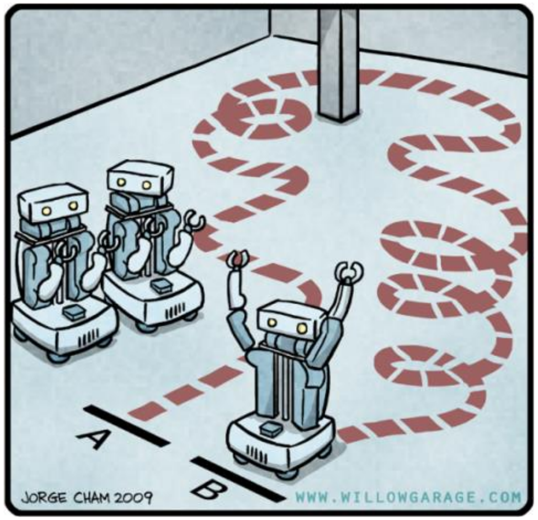
MT-RRT: a general purpose multithreading library for path planning

Andrea Casalino, Andrea Maria Zanchettin and Paolo Rocco

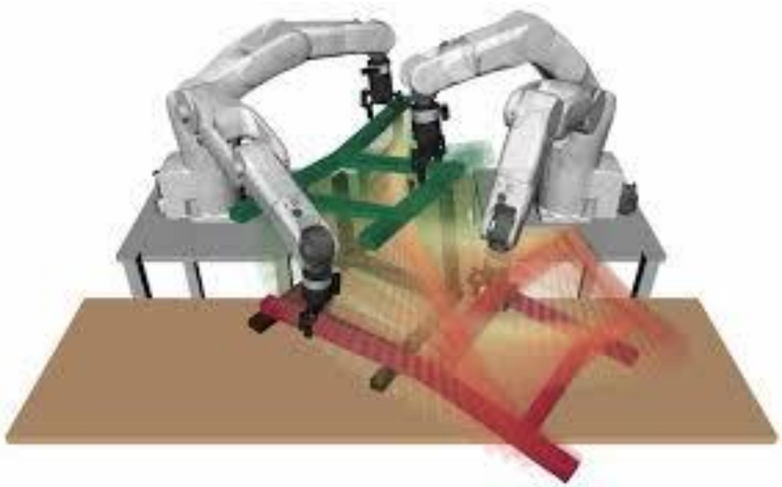
Motivations

Possible planning problems:

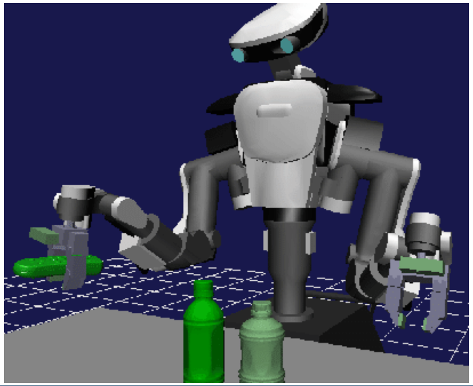
Mobile robots navigation



Pick and place



Plan grasping motions



Possible approaches

Possible approaches to solve a planning problem:

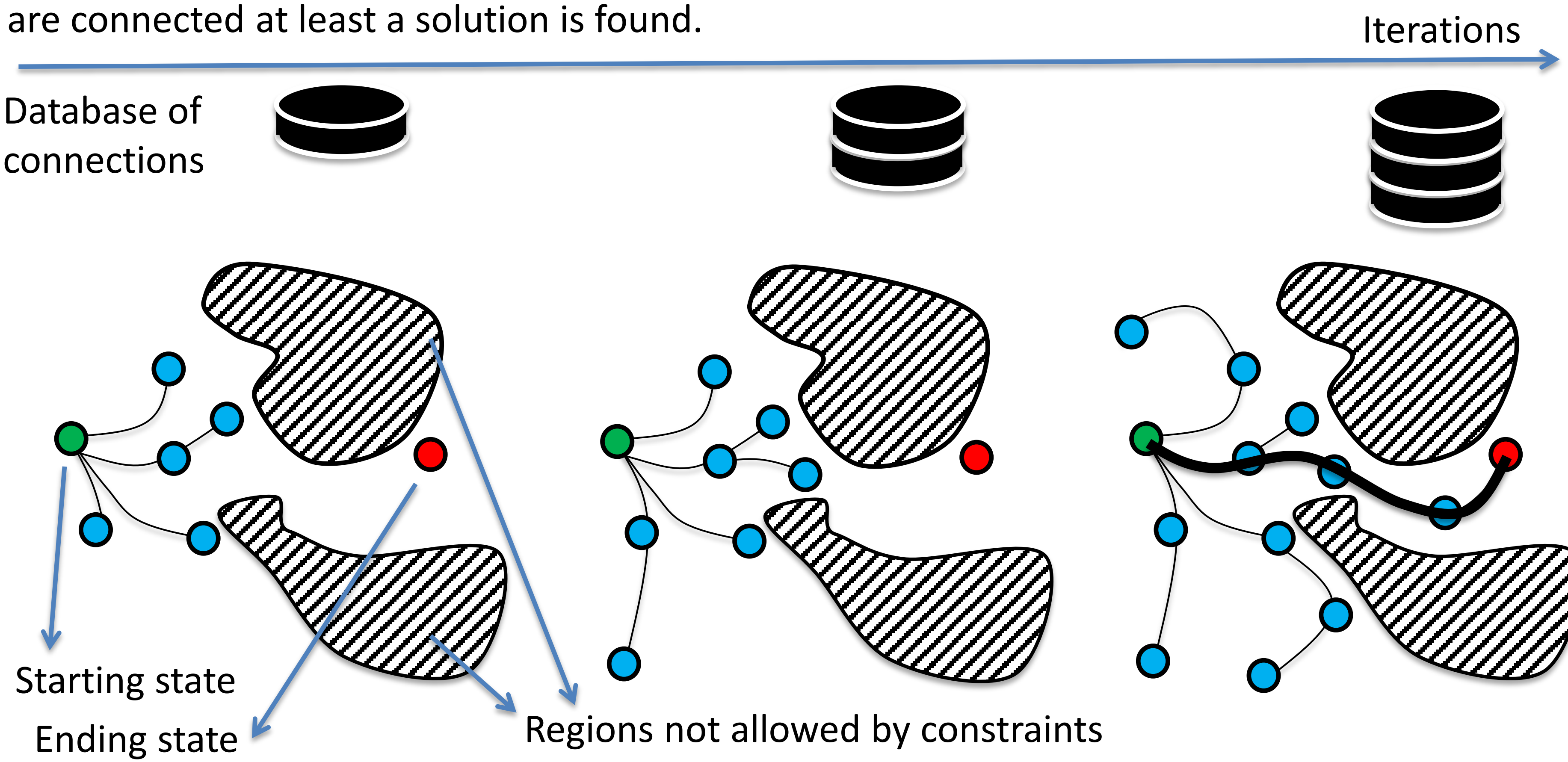
- ❑ closed loop control schemes
  - ❑ Repulsive/attractive fields → Local minimum problems
  - ❑ MPC approaches → May fail to find a feasible solution for complex workspace

- ❑ Rapidly random exploring tree strategies → Flexible and capable of always finding a solution, in the case a solution exists…

… but computionally intense: thousands of iterations are required also to get at a sub-optimal solution.

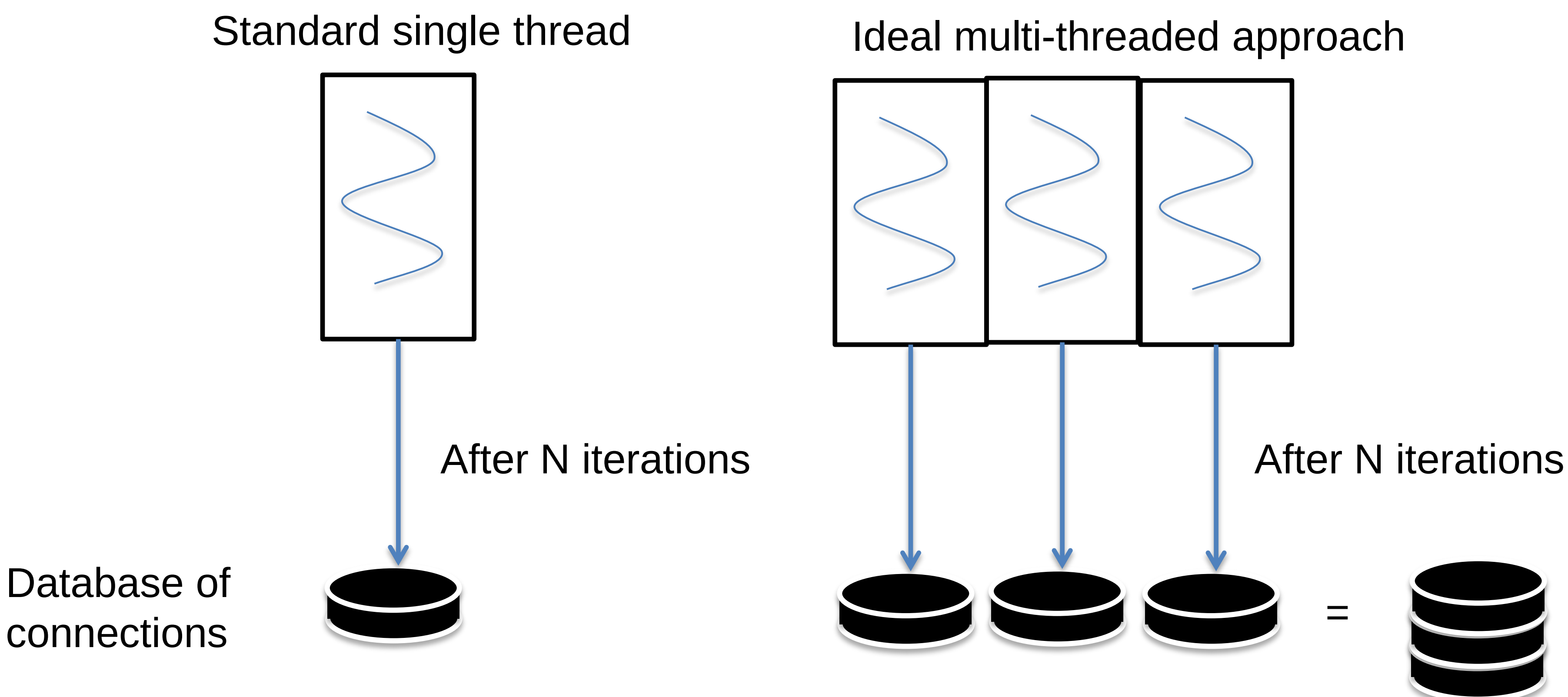
RRT mechanism

The algorithm consists essentially in exploring the configurational space admitted by a series of constraints, with the aim of building a search tree. Once the starting and the ending states are connected at least a solution is found.



Accelerating RRT algorithms

Why don't explore the configurational space in different threads in order to grow faster the search tree?



How to achieve a good parallelism when applying RRTs?

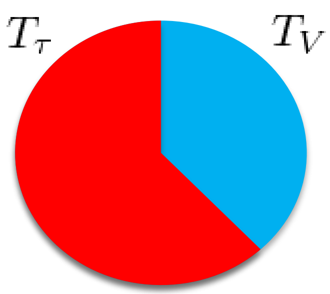
RRT profiling

The computational times are shared by the following main operations:

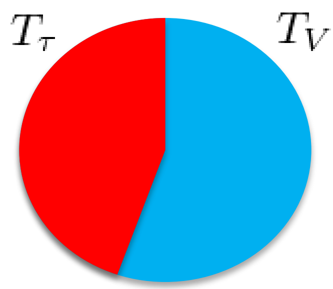
T_τ —— Optimal trajectory computation (includes Nearest Neighbour searches)

T_V —— Check the feasibility of sampled states

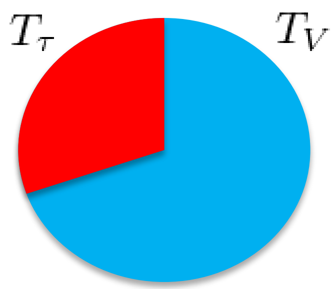
problem A



problem B



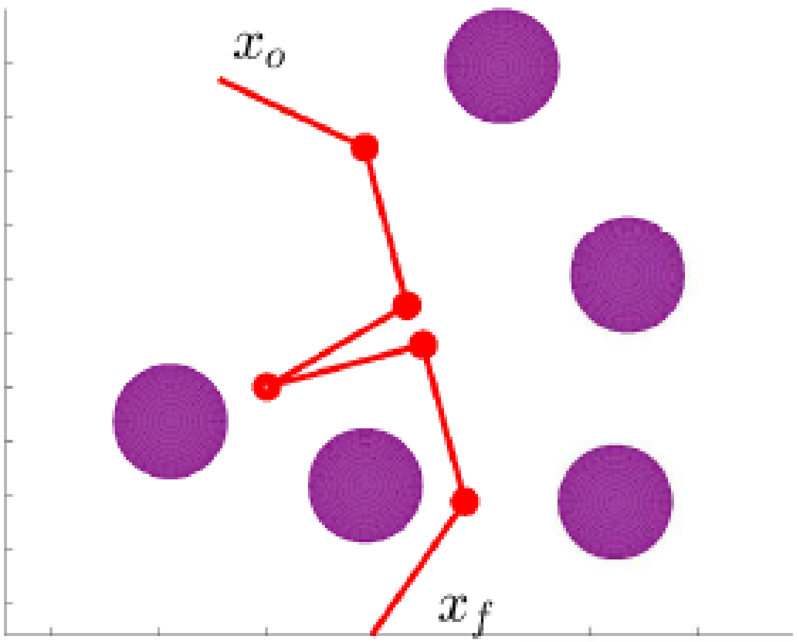
problem C



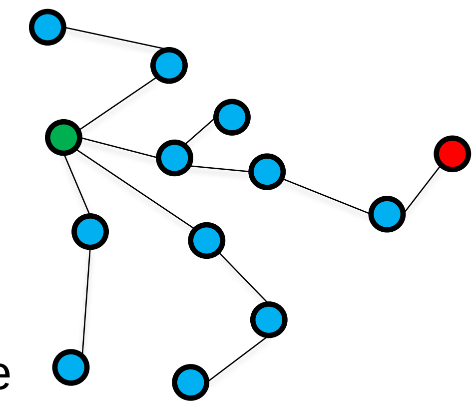
The ripartition of times can significantly vary from one application to another.

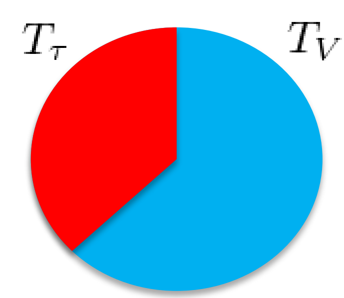
Benchmark A

Path planning of a planar 3 d.o.f. robot:



The optimal trajectories connecting the nodes are assumed as simple segments in the joint space



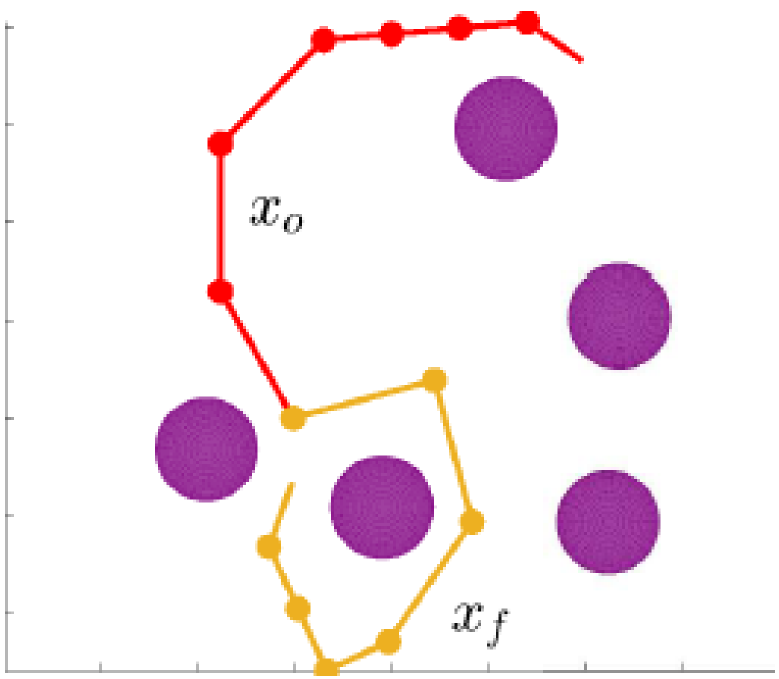


T_τ — Optimal trajectory computation (includes Nearest Neighbour searches)

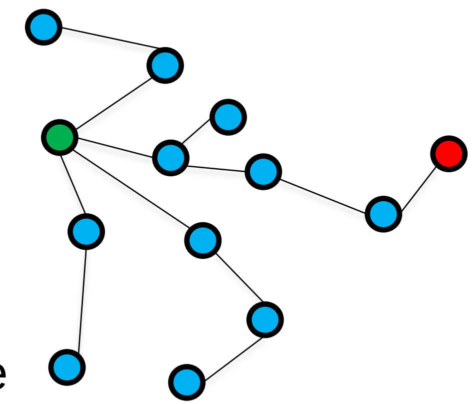
T_V — Check the feasibility of sampled states

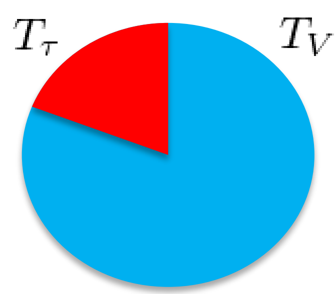
Benchmark B

Path planning of a planar 7 d.o.f. robot:



The optimal trajectories connecting the nodes are assumed as simple segments in the joint space



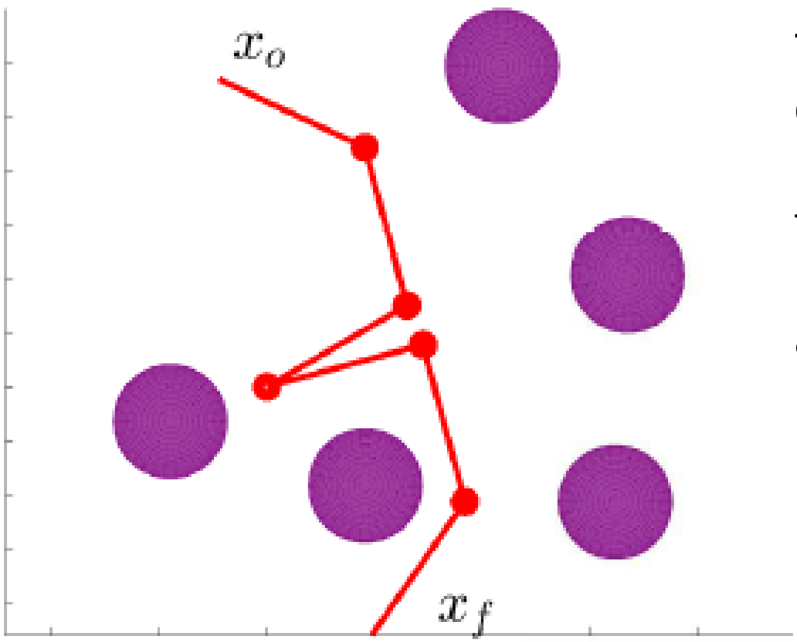


T_τ — Optimal trajectory computation (includes Nearest Neighbour searches)

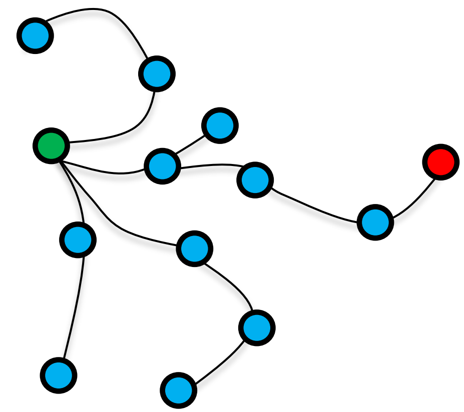
T_V — Check the feasibility of sampled states

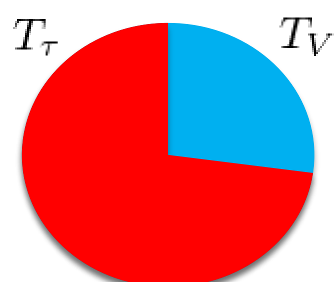
Benchmark C

Trajectory planning of a planar 3 d.o.f. robot:



The optimal trajectories connecting the nodes are minimum time second order time laws. The state is made of both the position and the velocity of the robot





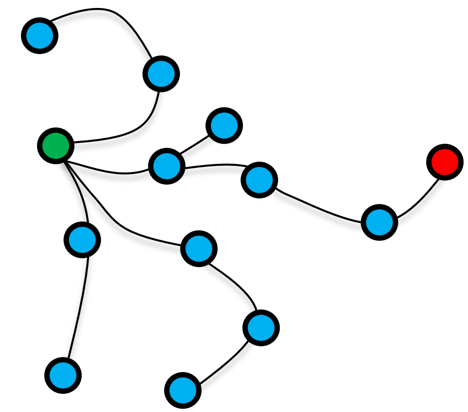
T_τ Optimal trajectory computation (includes Nearest Neighbour searches)

T_V Check the feasibility of sampled states

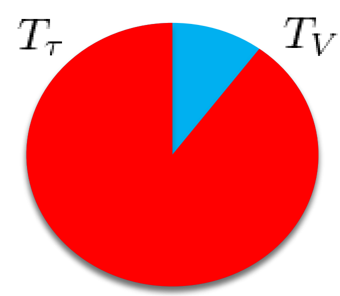
Benchmark D

Kinodynamic planning:

$$
\begin{cases}
\dot{x} = f(x) + u \\
u = u_o + \left.\frac{\partial f}{\partial x}\right|_{x_o} \left(x - x_o \right) + K \left(x_f - x \right) \\
s.t. \quad x \in \mathcal{X}, u \in \mathcal{U}
\end{cases}
$$



The optimal trajectories connecting the nodes are computed by considering a LQR controller applied on a linearization of the system.



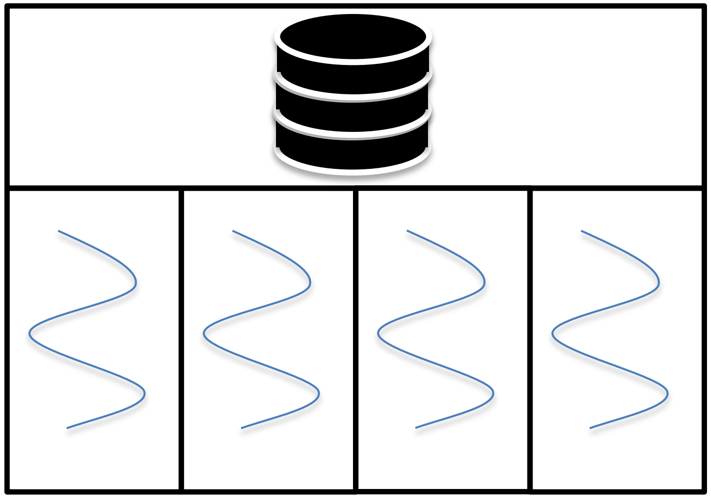
T_τ —— Optimal trajectory computation (includes Nearest Neighbour searches)

T_V —— Check the feasibility of sampled states

Shared tree exploration

The whole exploration process is parallelized. Threads share the same tree and have to synchronize each other for modifying it.

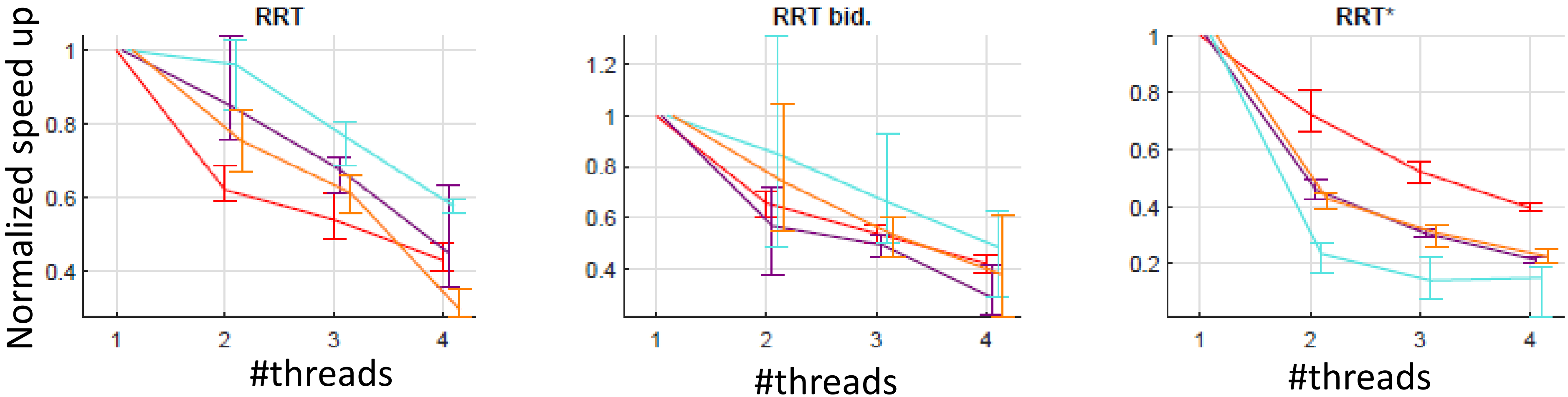
Shared search tree



Exploring threads

Shared tree exploration: results

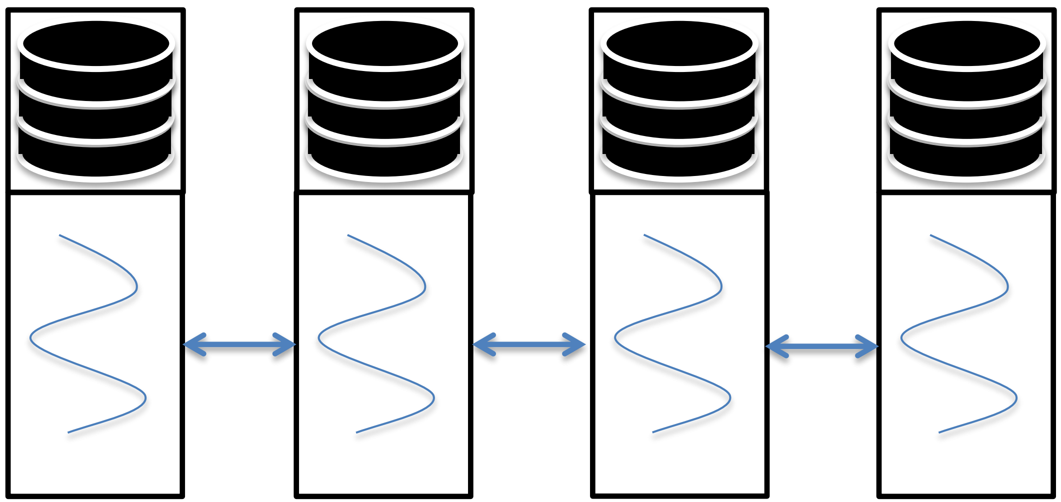
3 dof path planning | 7 dof path planning | 3 dof traj. planning | Kinodynamic planning



Local tree copies

The whole exploration process is parallelized. Each thread has its proper tree to expand. In this case, threads don't need to synchronize at each iteration. With a certain frequency, data are exchanged among threads.

The trees owned by threads



Exploring threads

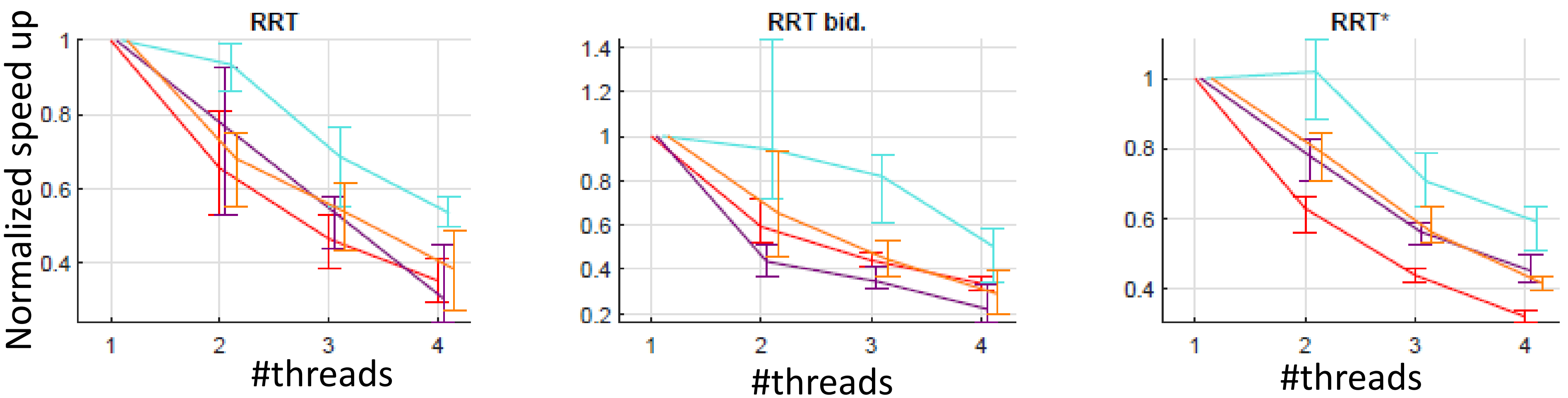
Local tree copies: results

3 dof path planning

7 dof path planning

3 dof traj. planning

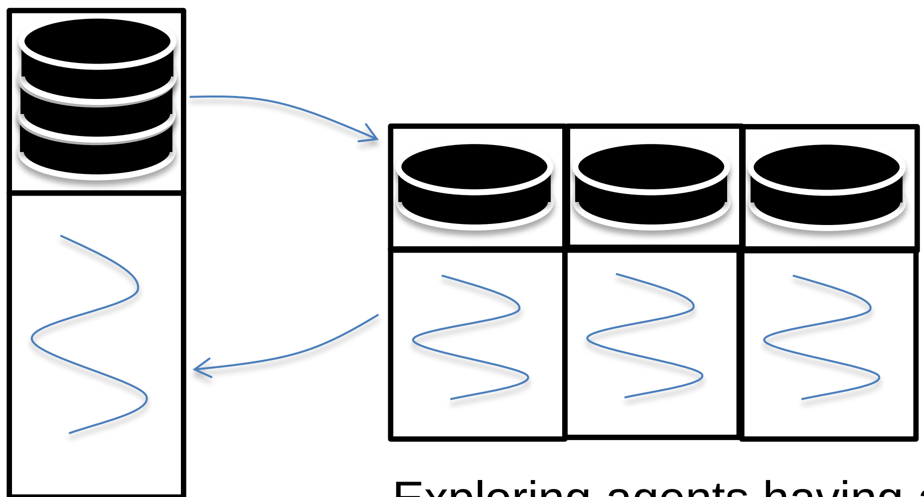
Kinodynamic planning



Multi agents approach

Blinded multi agents exploration: a pool of threads locally explore the configurational space, starting every time from a new root and obtaining a new tree. The explorations are notified to a master, which integrates the knowledge acquired by the slaves into a single tree and dispatch new roots to explore.

Rewirds are locally done on the local tree in case of the RRT*.

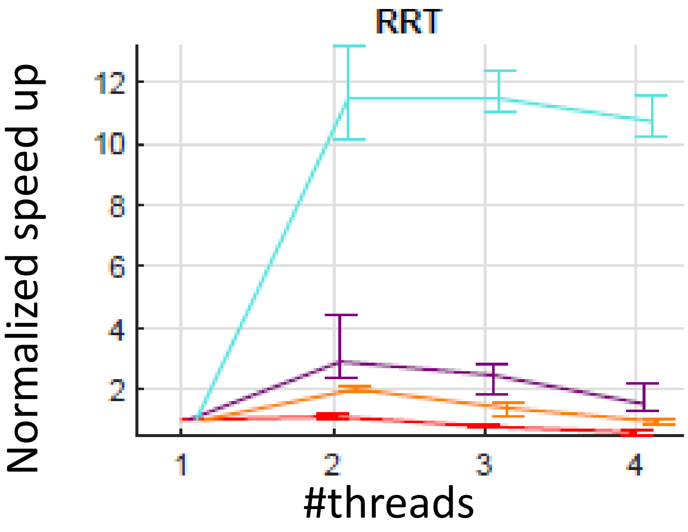


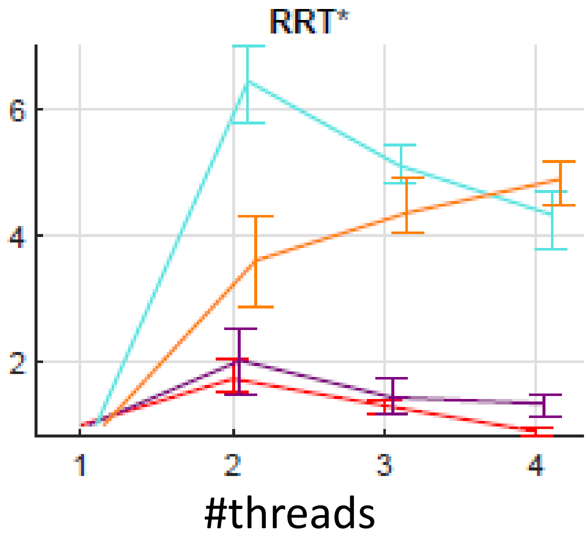
Exploring agents having a local knowledge

Master thread

Multi agents approach: results

3 dof path planning | 7 dof path planning | 3 dof traj. planning | Kinodynamic planning



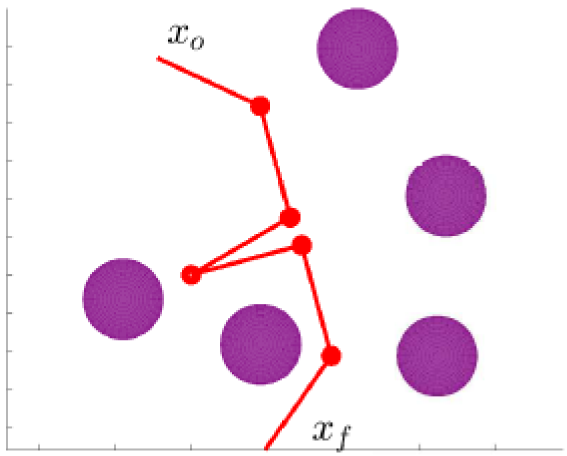


Speed up increased dramatically

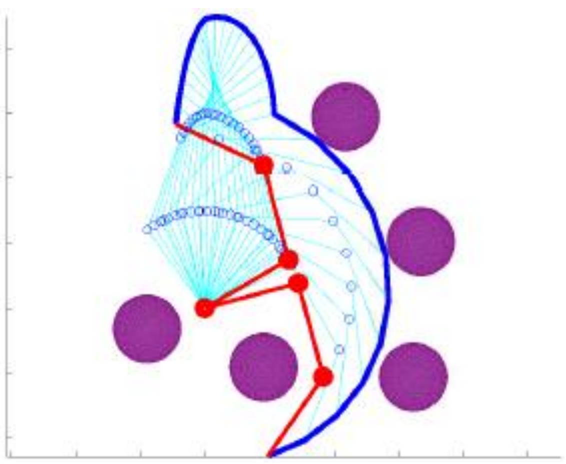
Multi agents approach

The optimality property of the RRT* seems to be preserved also in this multi agents strategy:

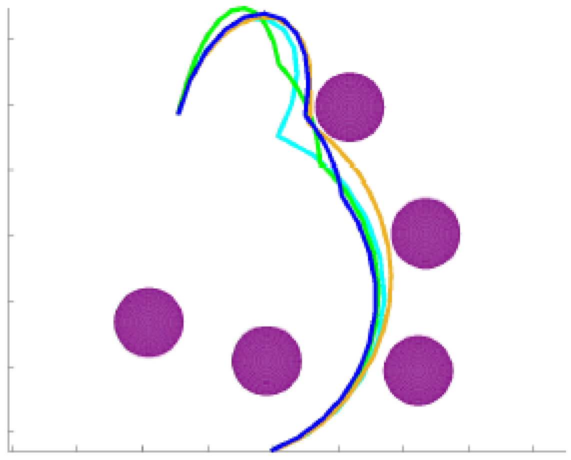
3 d.o.f. path planning problem



Minimum distance path (in joint space)



Solutions obtained by the RRT* parallelized with the multi agents strategy



MT-RRT C++ library

All the proposed strategies are contained in an open source library C++. It was made so as to easily customize a new planning problem and use the proposed solvers.

Check out the the Github profile:

https://github.com/andreacasalino/MT_RRT