Real-time Motion Planning with Dynamic Obstacles

Jarad Cannon, Kevin Rose, and Wheeler Ruml

UNIVERSITY of NEW HAMPSHIRE

(thanks to the NSF RI and DARPA CSSG programs for support)

Motivation

- planning for robots!
- responsiveness, safety: real-time with dynamic 'obstacles'
- kino-dynamic motion planning, not shortest path
- heuristic search is general and powerful
- human environments are generally forgiving

Given at start:

- map of static world
- goal pose for agent
- time bound per action

Given at each step:

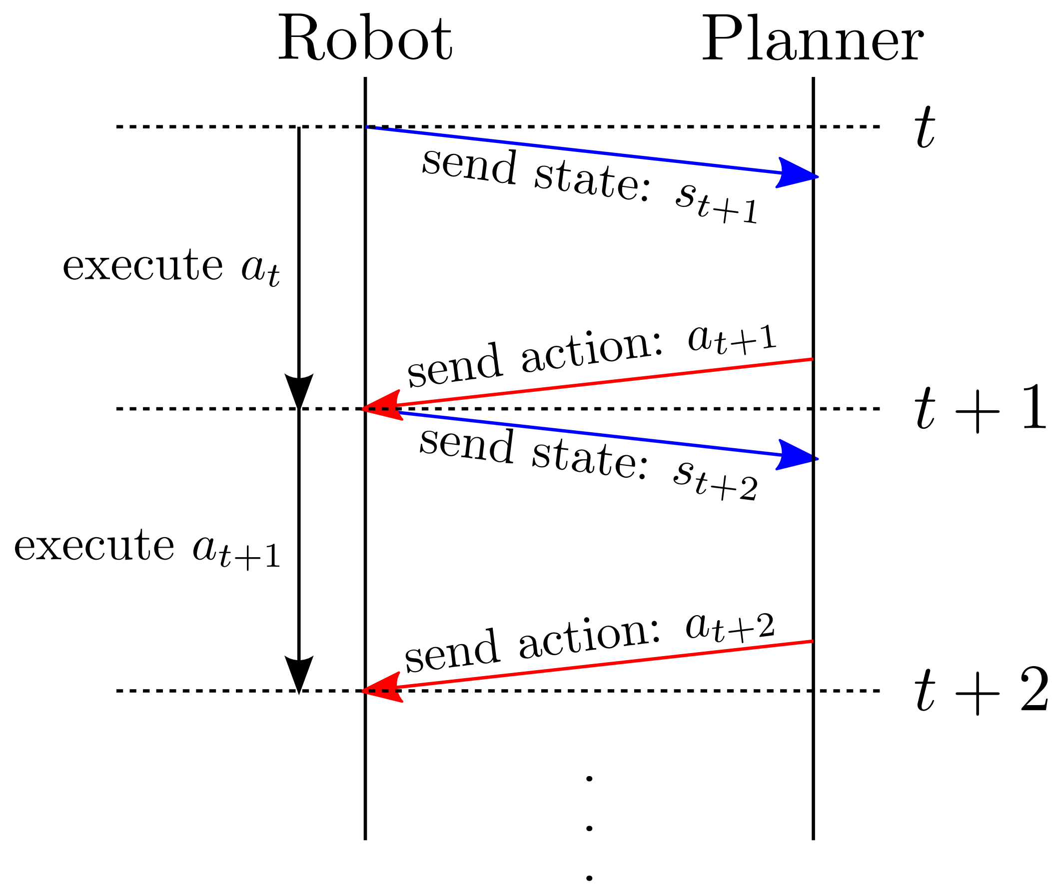
- agent's estimated pose
- probabilistic estimates for each dynamic obstacle's future

Find:

- within time bound
- feasible action
- that minimizes agent's total cost to horizon

Execution Setting

while executing, plan for next action



Heuristic Search

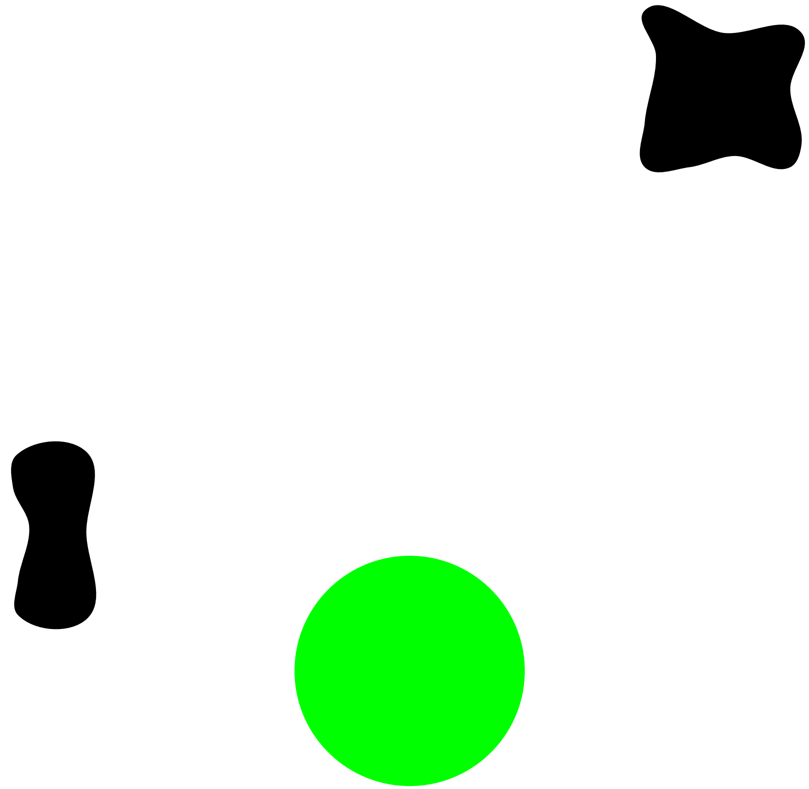
agent's state: x, y, θ, s

obstacles move, so world state adds t

discretize space and time to yield motion primitives

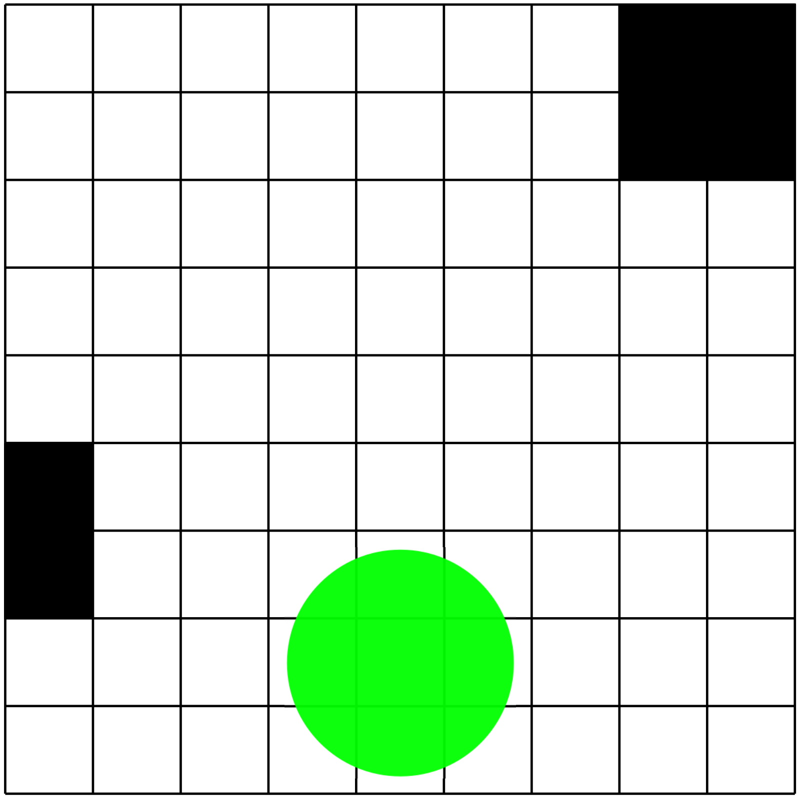
cost$= P(col) \cdot cost_{col} + cost_{time}$

The Search Space



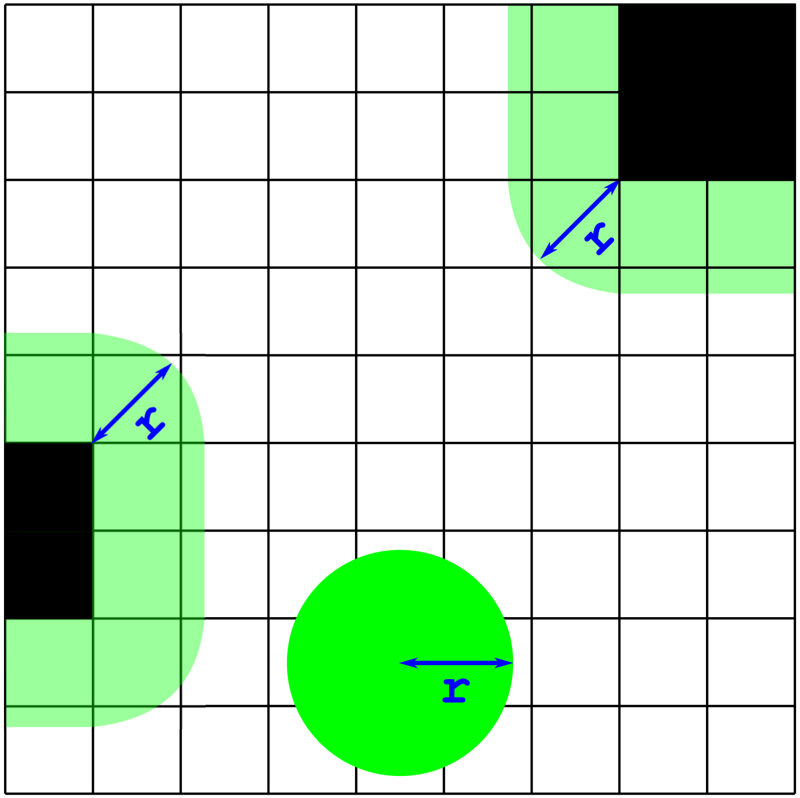
robot with two static obstacles

The Search Space



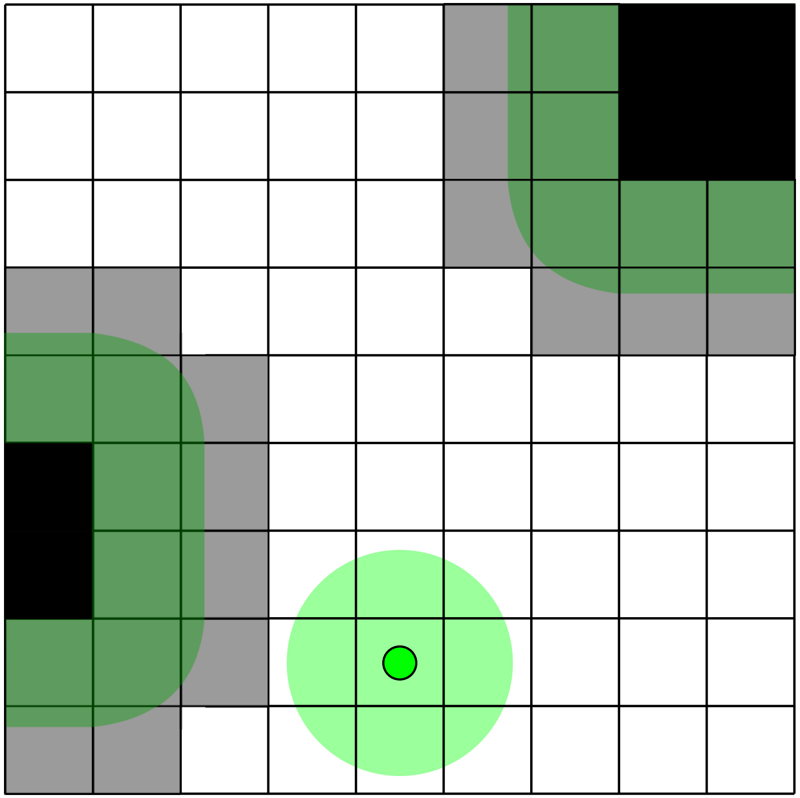
overlay grid

The Search Space



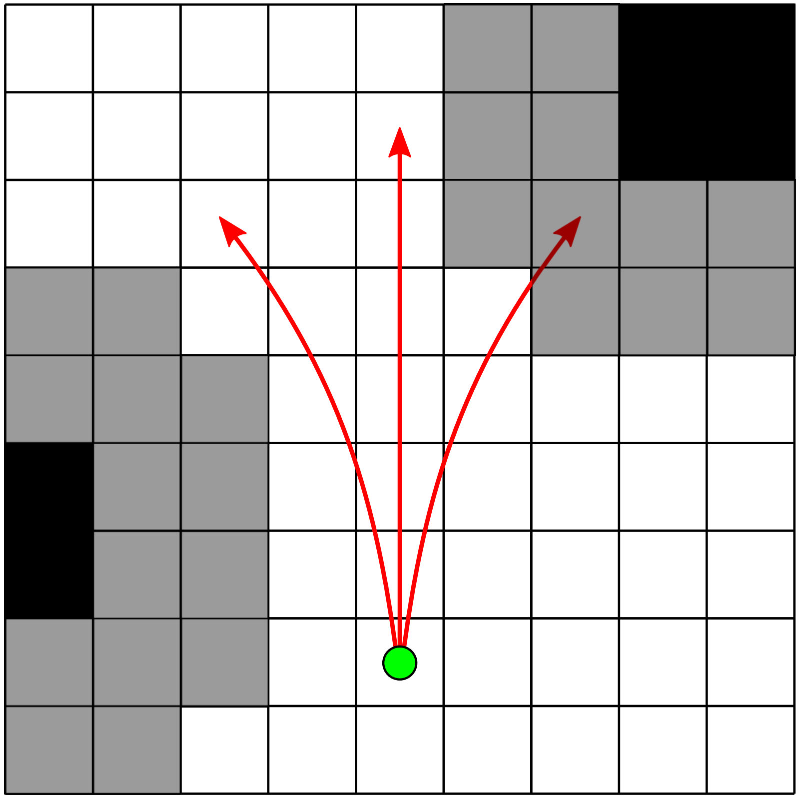
expand static obstacles by robot's radius

The Search Space



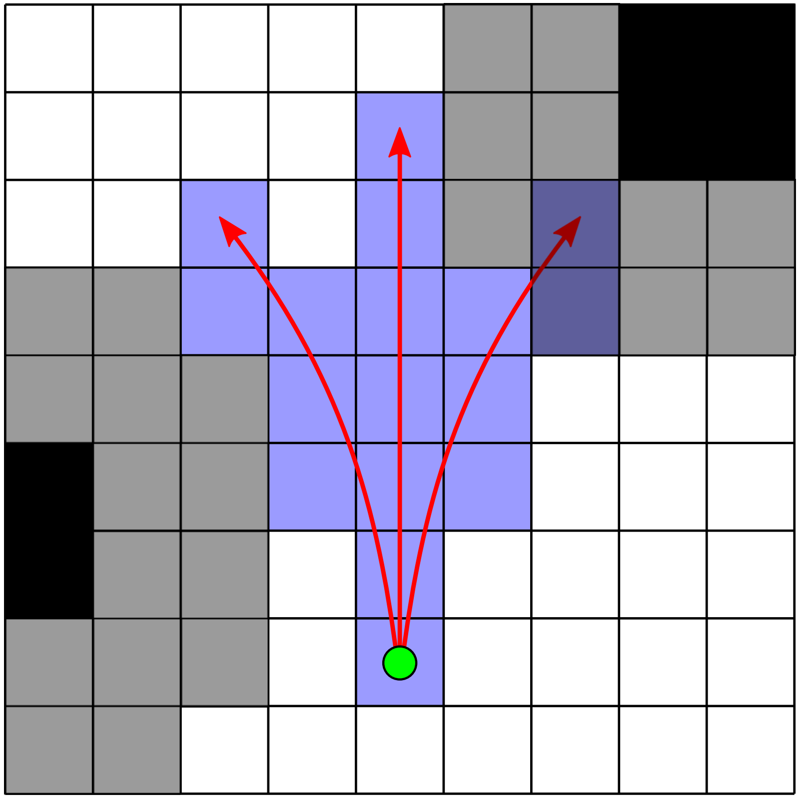
robot now a single point

The Search Space



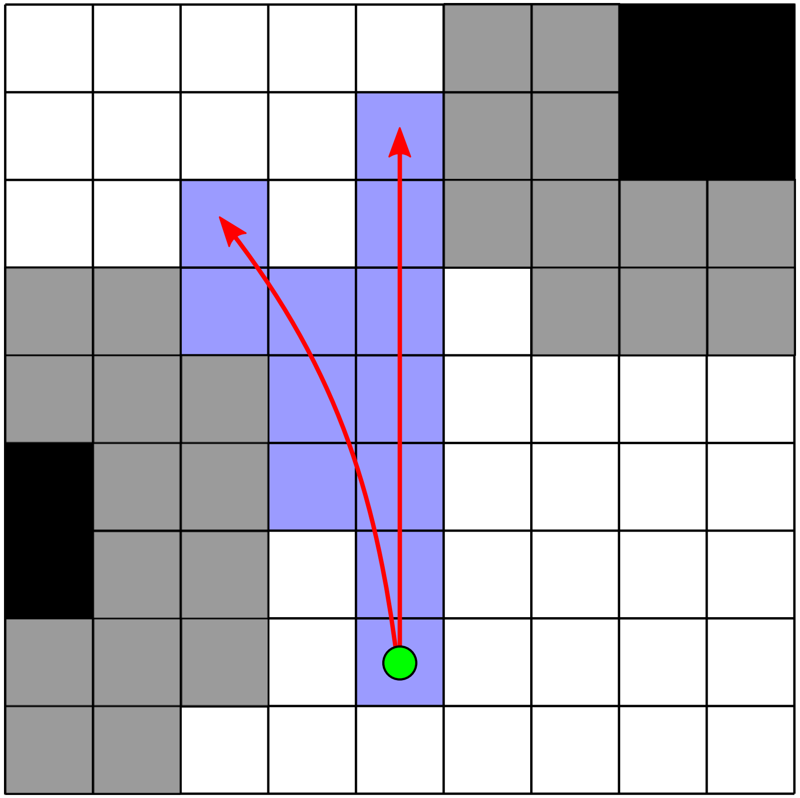
applicable actions

The Search Space



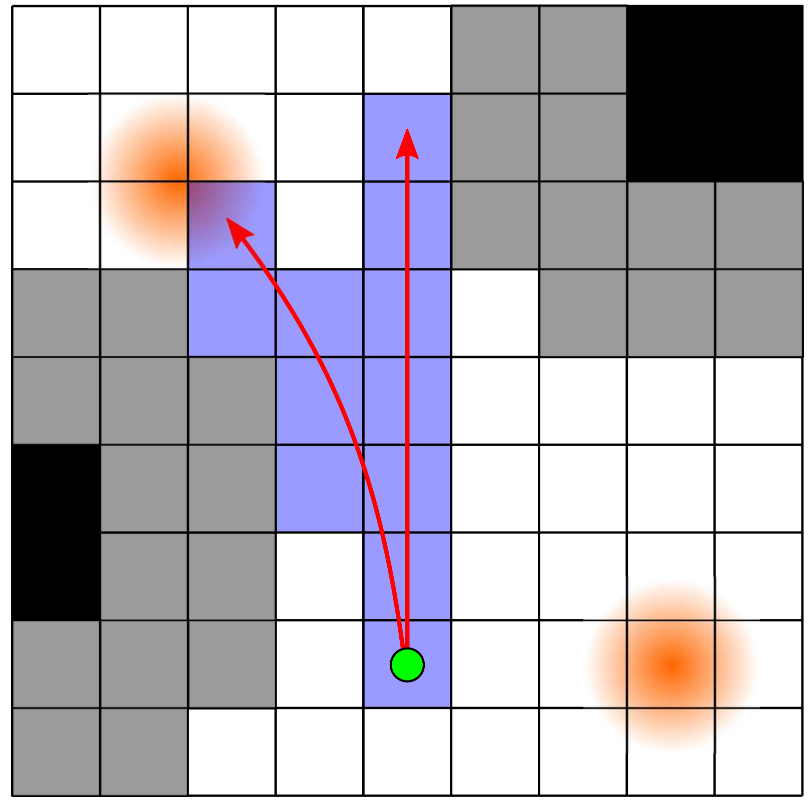
(pre-)compute touched cells

The Search Space



remove actions with static collisions

The Search Space



time-dependent obstacle probability distributions

Two Approaches

Motion Planning:

Take a search algorithm designed for motion planning, and make it real-time.

Real-time Search:

Take a real-time search algorithm, and adapt it to motion planning.

The Heuristic Search Literature

Motion Planning:

"R* [provides] probabilistic guarantees on the suboptimality of the overall solution [and] can scale to large complex planning problems, can find solutions to such problems much more often than weighted A* search, and can minimize the cost of the found solutions much better than randomized motion planning algorithms, developed specifically for continuous domains." — Likhachev and Stentz, *AAAI*, 2008

Real-time Search:

"We illustrate agent-centered search in nondeterministic domains using robot-navigation tasks" — Koenig, *AI Magazine*, 2001

"State-of-the-art real-time search algorithms, like LSS-LRTA*" — Hernandez ands Baier, *JAIR*, 2012

Real-time R*

- Combines ideas from heuristic search and random sampling
- Breaks search up into smaller, easier subproblem
- High-level nodes are sampled randomly
- Low-level search between high-level nodes

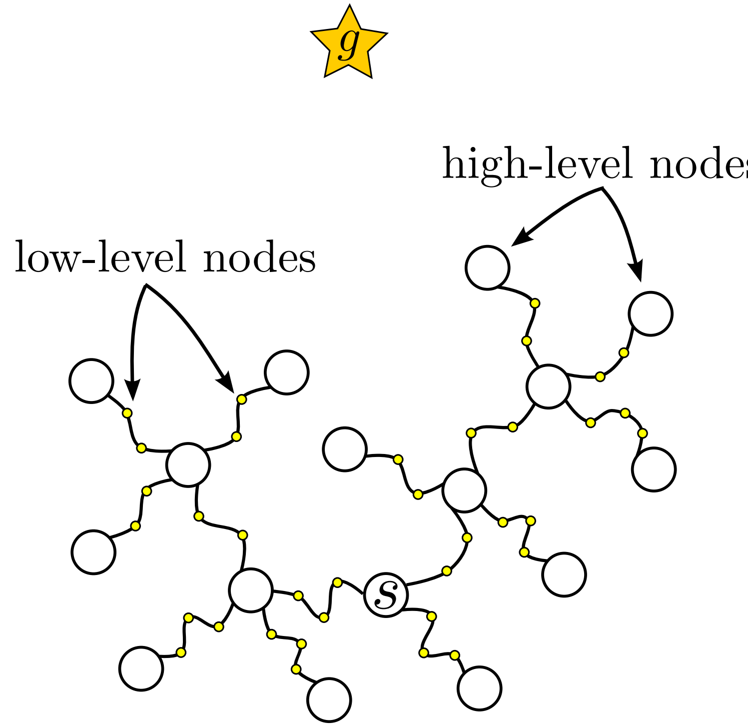


Figure 1: R* Search

(Likhachev and Stentz, 2008)

R* Advantages

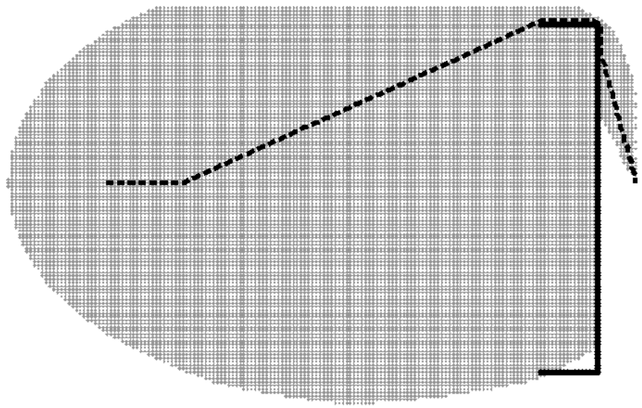


Figure 2: A* Search

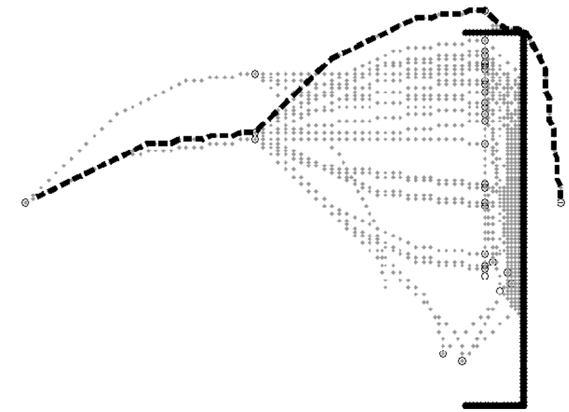


Figure 3: R* Search

- samples continuous space sparsely
- finds solutions faster than ARA*, lower cost than RRT
- works well in high dimensions and with local minima

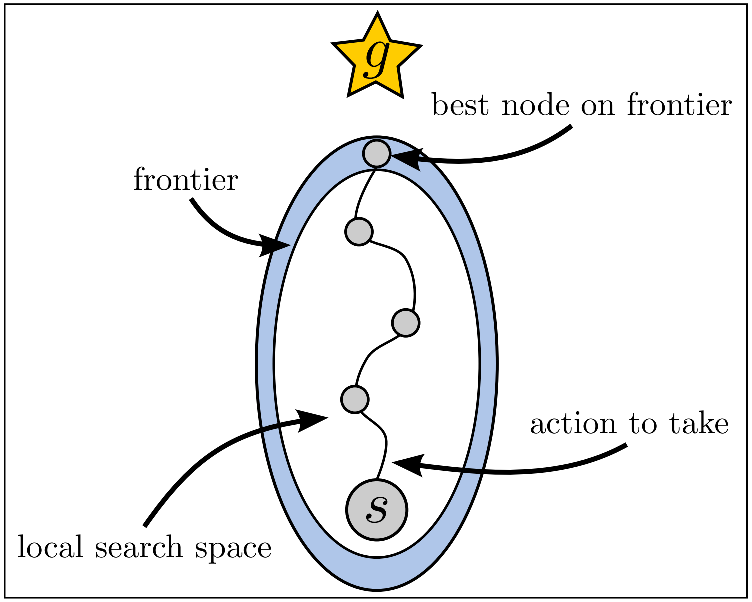
Modifications:

1. Bounded lookahead: count high-level expansions more
2. Action selection: prefer nodes with low-level paths
3. Path caching: save high-level nodes only
4. Making easier subproblems: loose goal test
5. Limit work on “avoid” nodes: exponentially increasing bound

see paper for details

RTA* for Robotics

- A* until node expansion limit.
- Move toward best node on *open*



- Learn improved heuristic for all expanded nodes
  - Compute shortest paths back from the frontier

(Koenig and Sun, 2009)

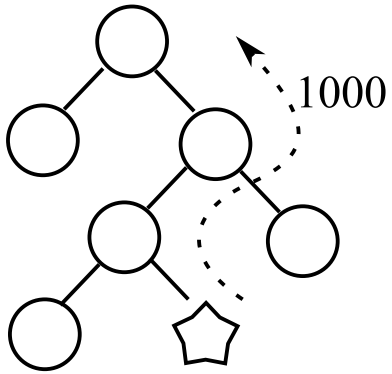
The Challenge of Dynamic Motion Planning

theoretical: inadmissible g values

- no good h for dynamic obstacles
- estimated collisions noticed via g values
- edge costs in graph are changing!
- backed-up values inadmissible

practical: many possible paths

- hard to back-up a higher value



Partitioned Costs

Modifications:

- partition cost into static versus dynamic

$$f(n) = g_s(n) + g_d(n) + h_s(n) + h_d(n)$$

- learn separate static and dynamic h values
- all states at pose share h_s value
- decay h_d values for completeness

see paper for details

Evaluation

Evaluation

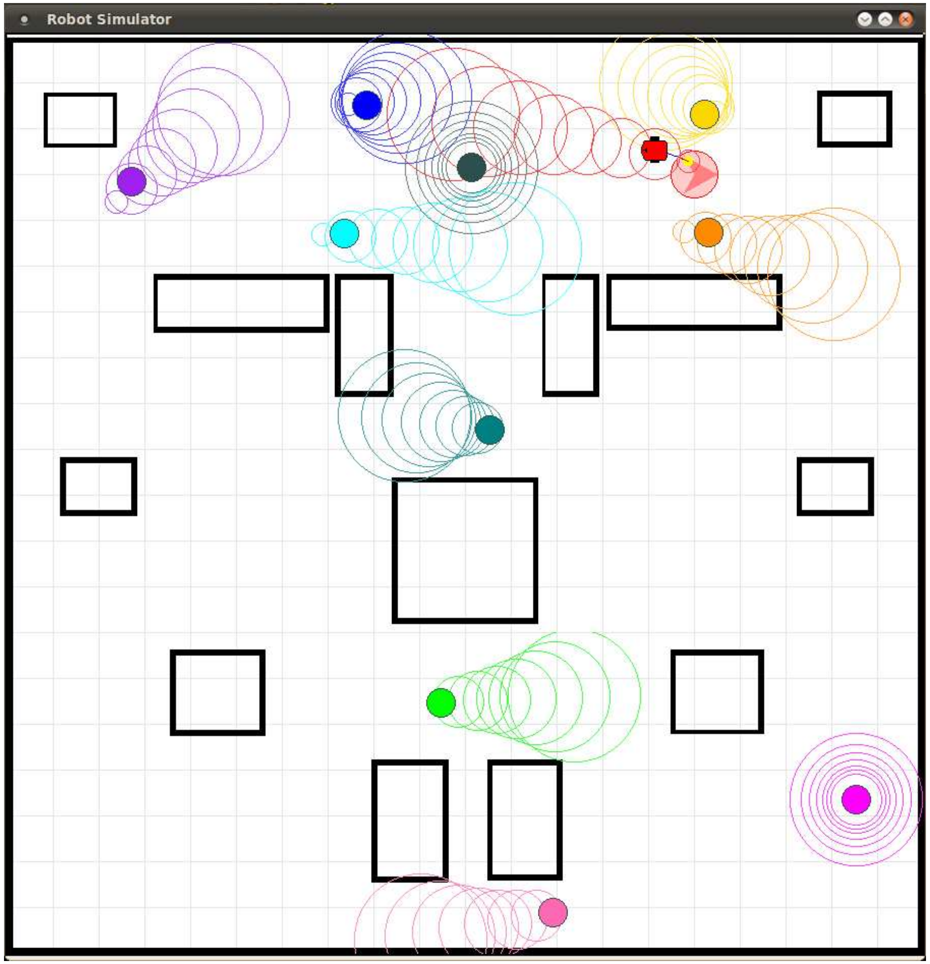
unusually wide variety of algorithms:

1. RTA*
2. LSS-LRTA*
3. PLRTA* - partitioned learning
4. R*
5. RTR* - real-time
6. RRT
7. TBL

two types of problems:

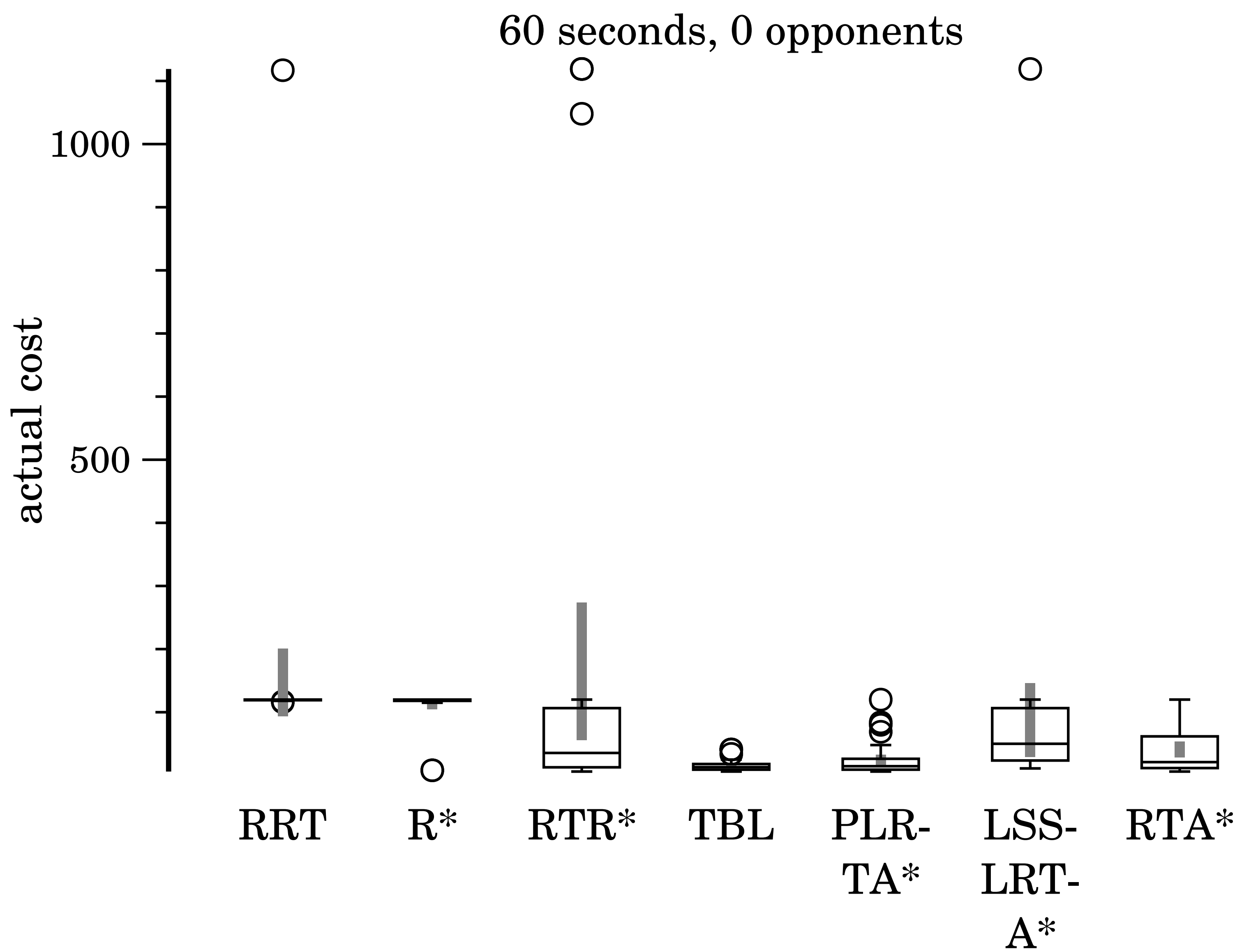
1. 36 random ‘frogger’ benchmarks
2. 6 handcrafted challenge problems

Random Benchmarks

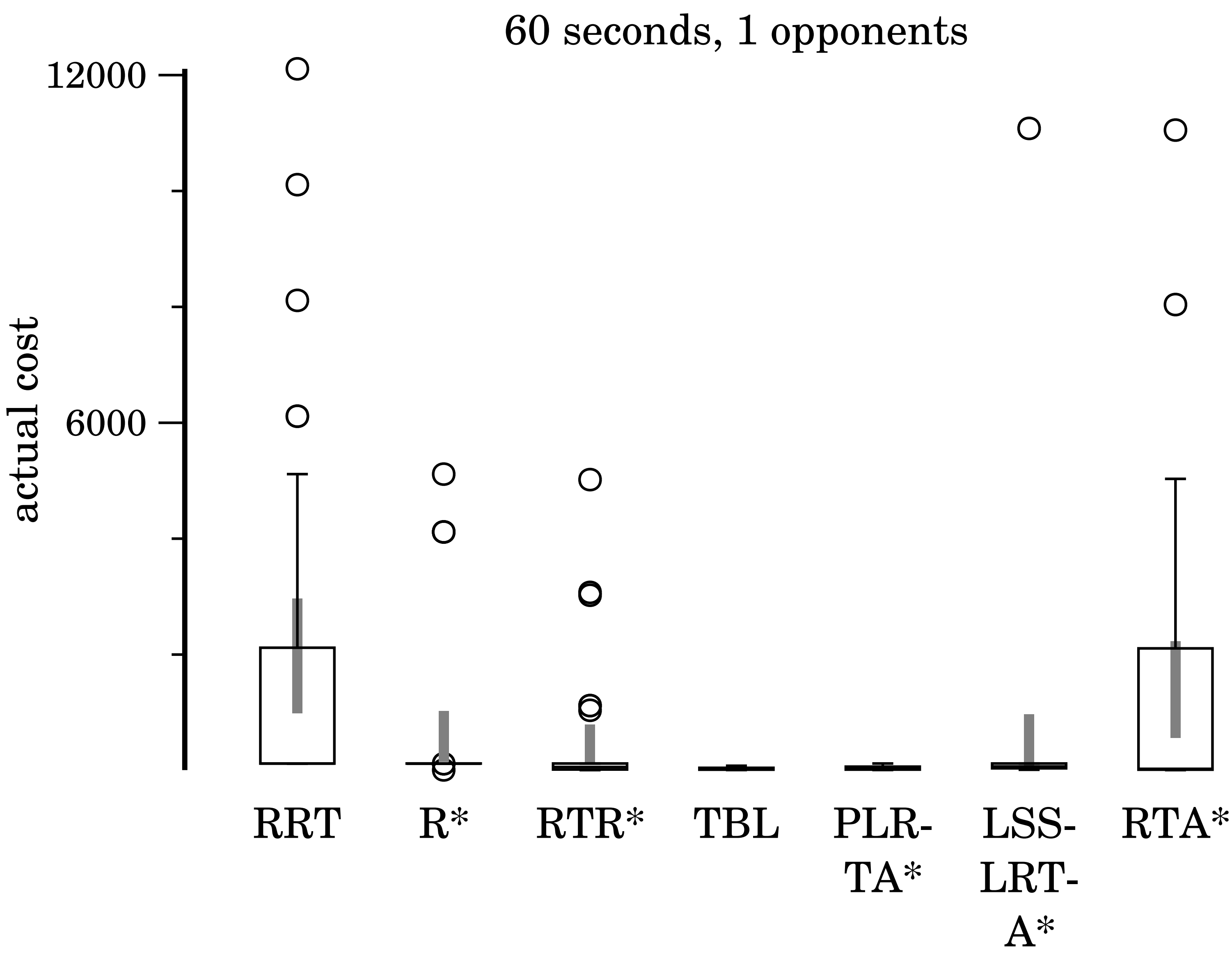


36 random start/goal pairs, with prerecorded obstacles

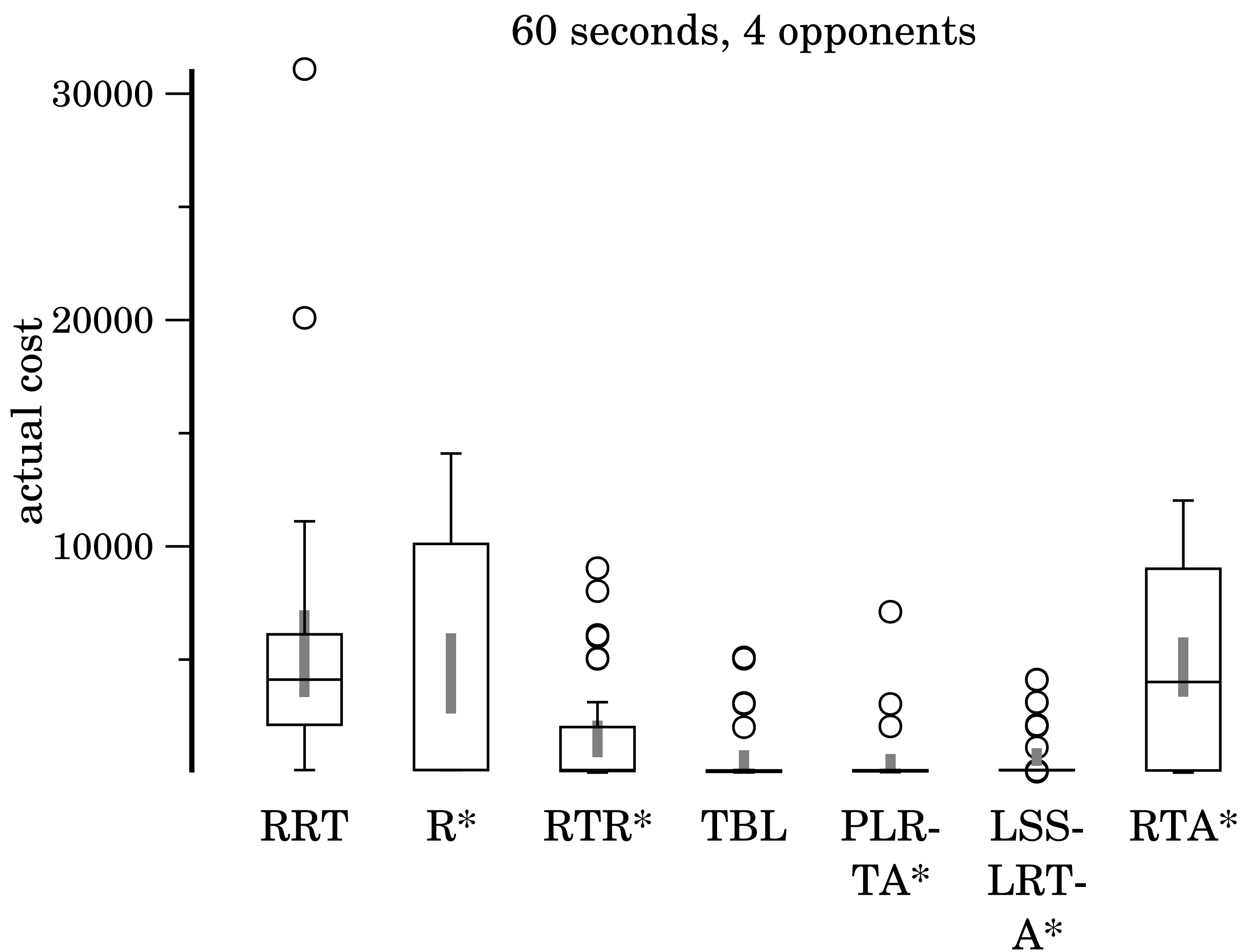
Cost Incurred



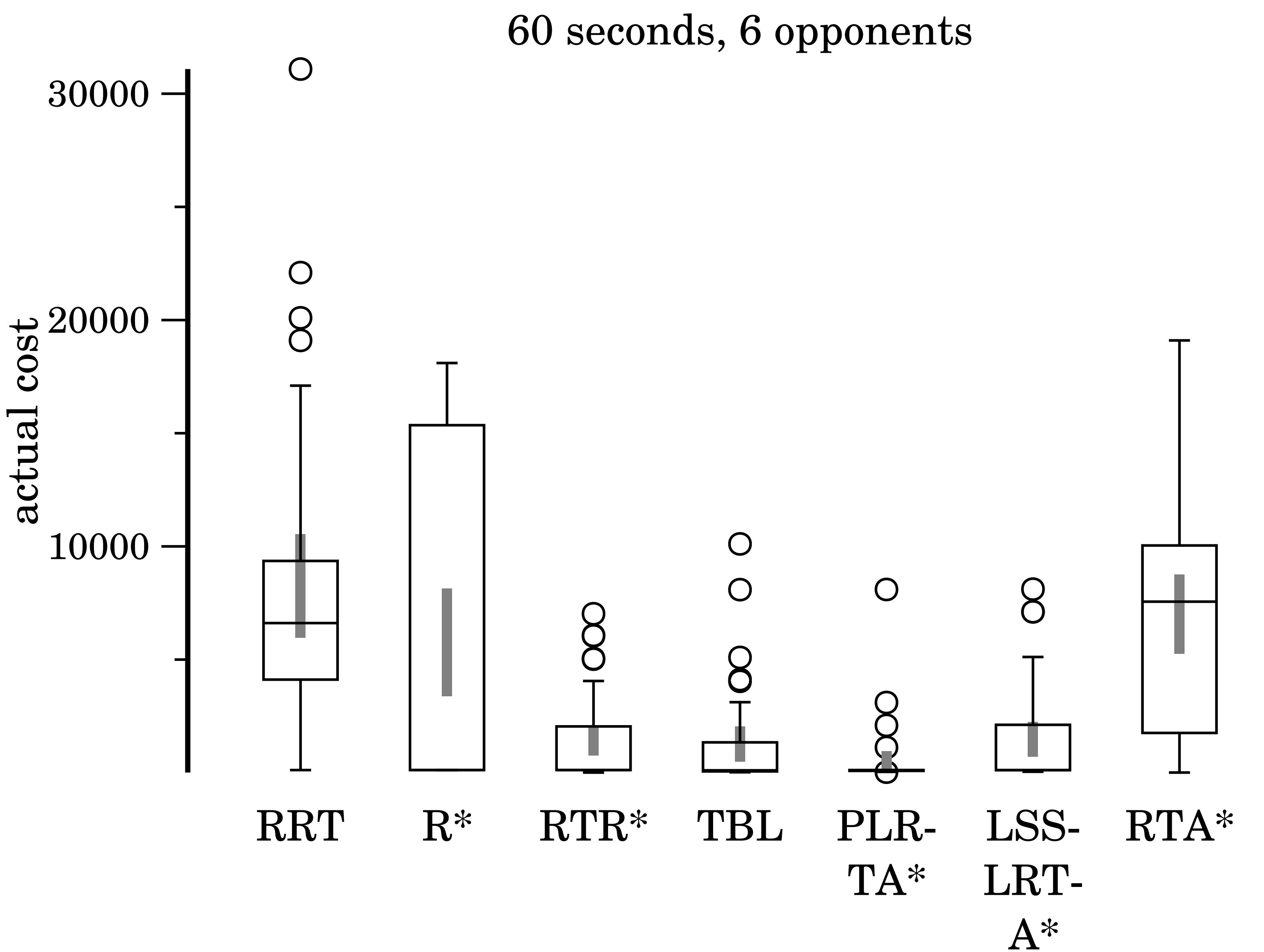
Cost Incurred

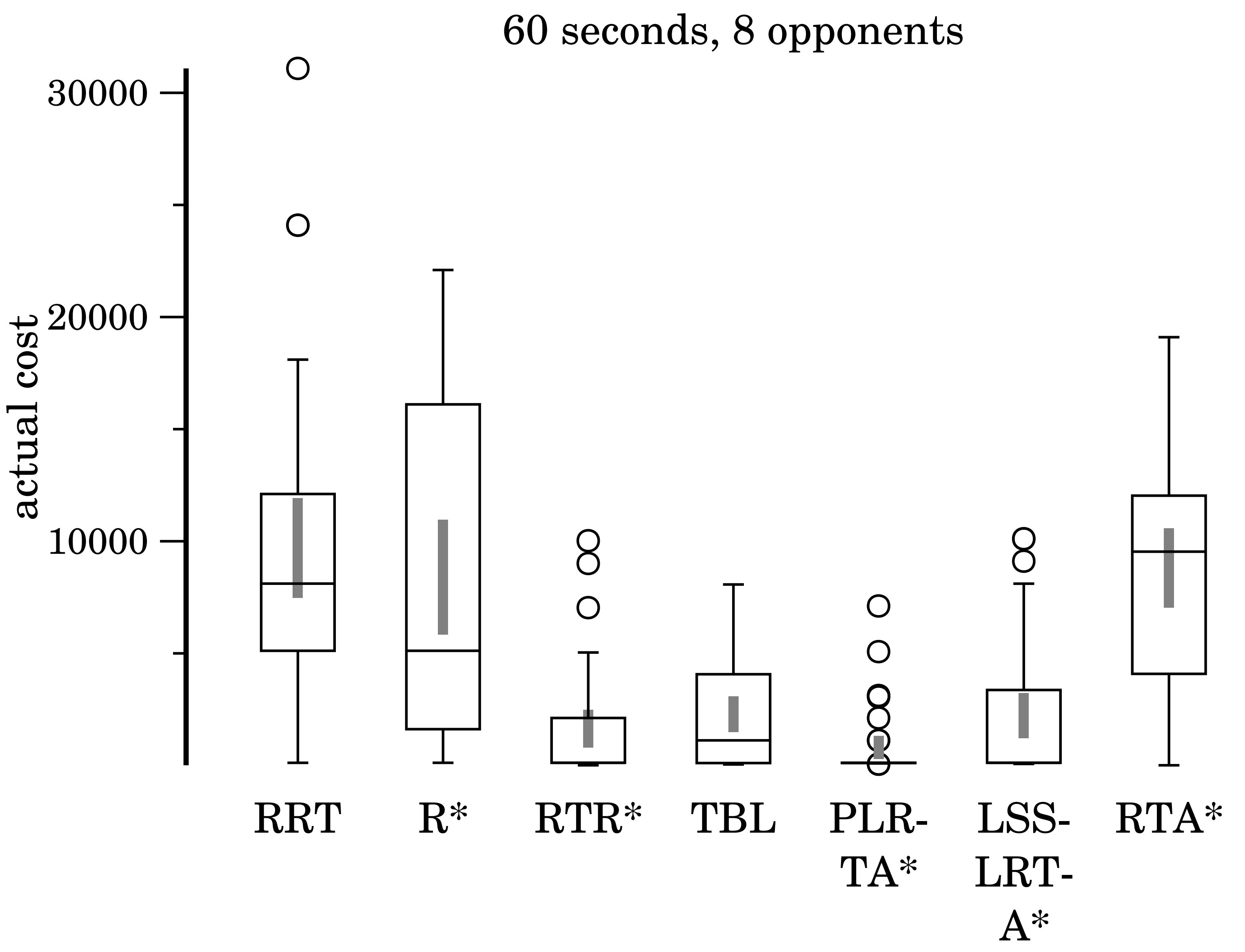


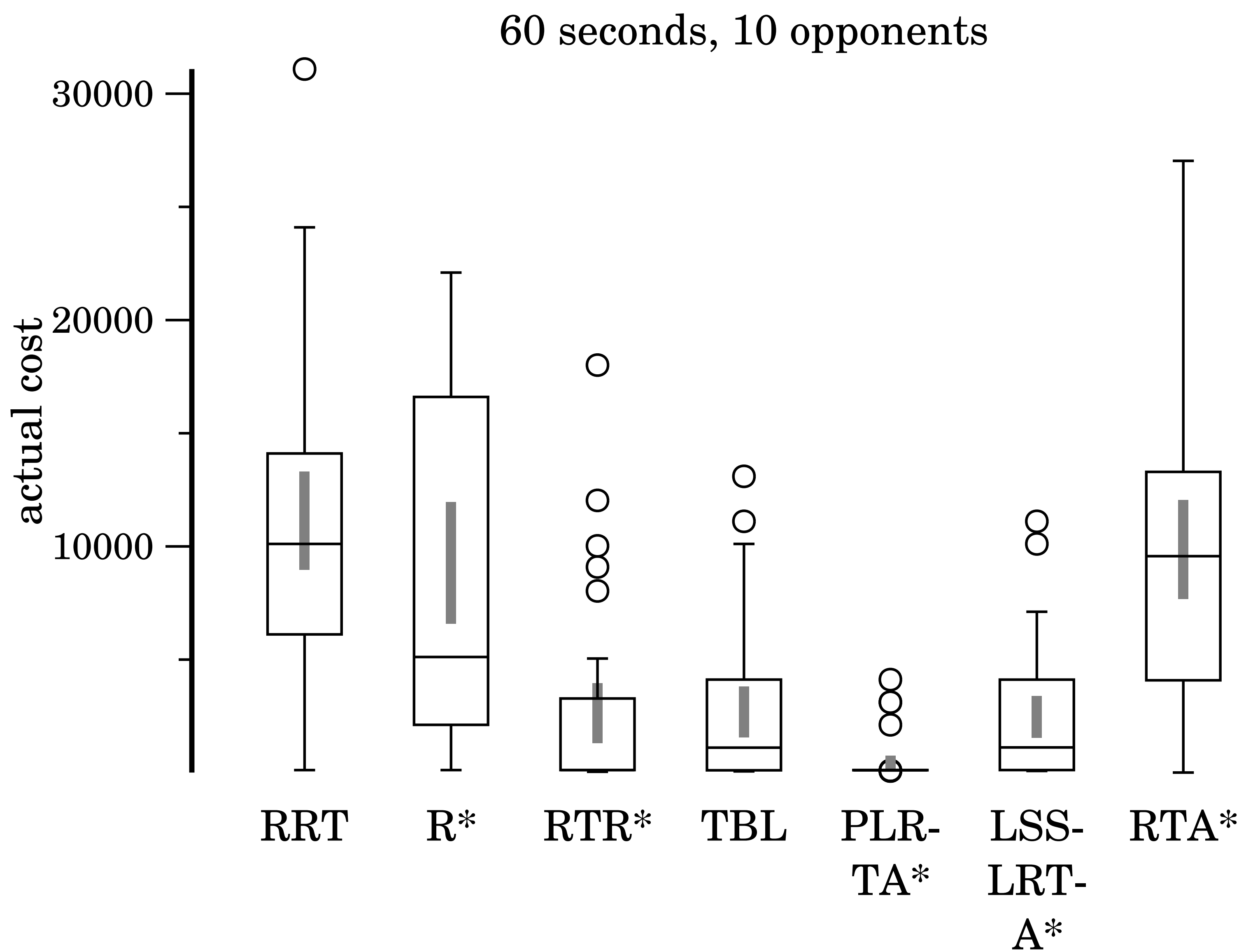
Cost Incurred

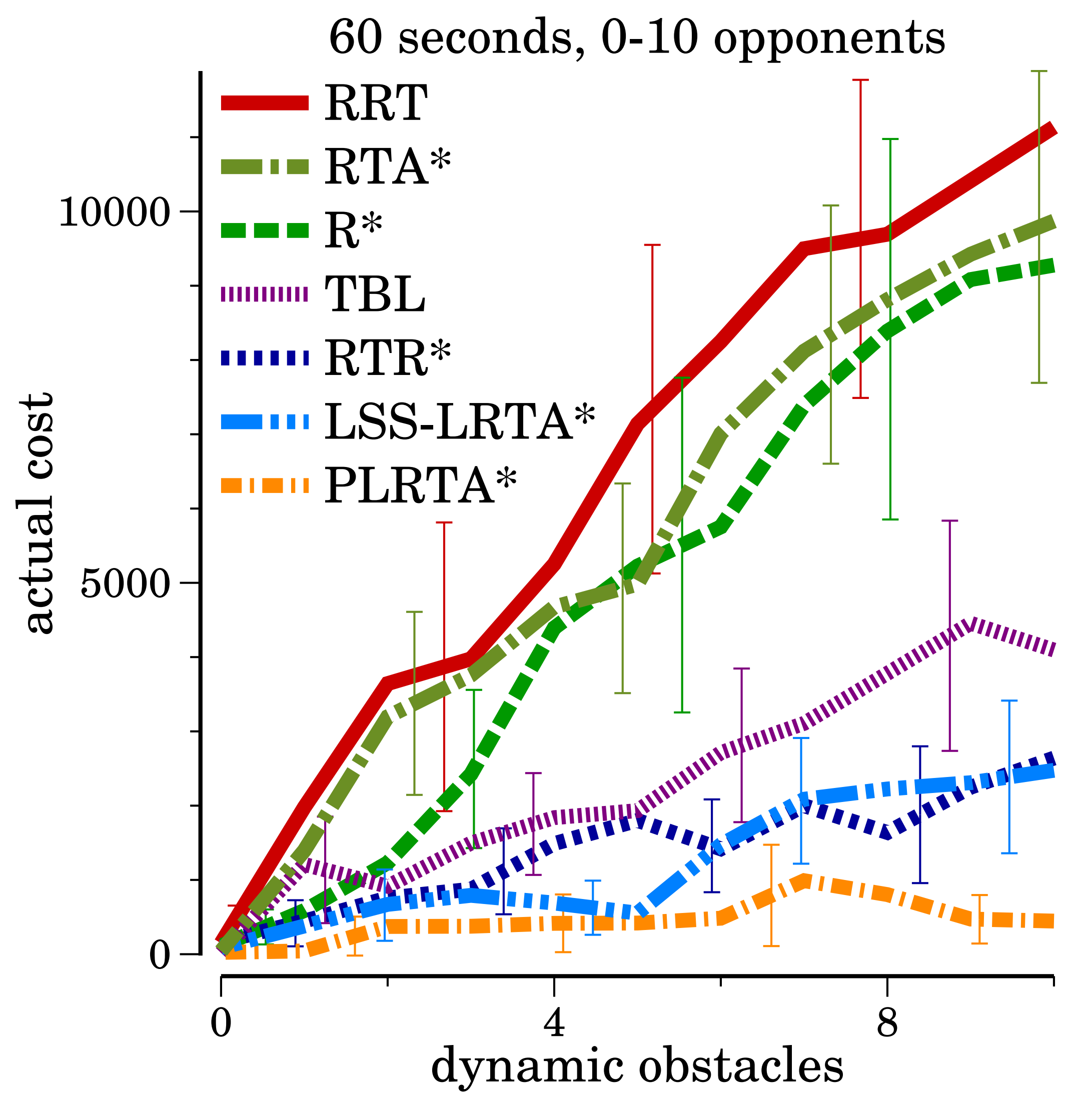


Cost Incurred



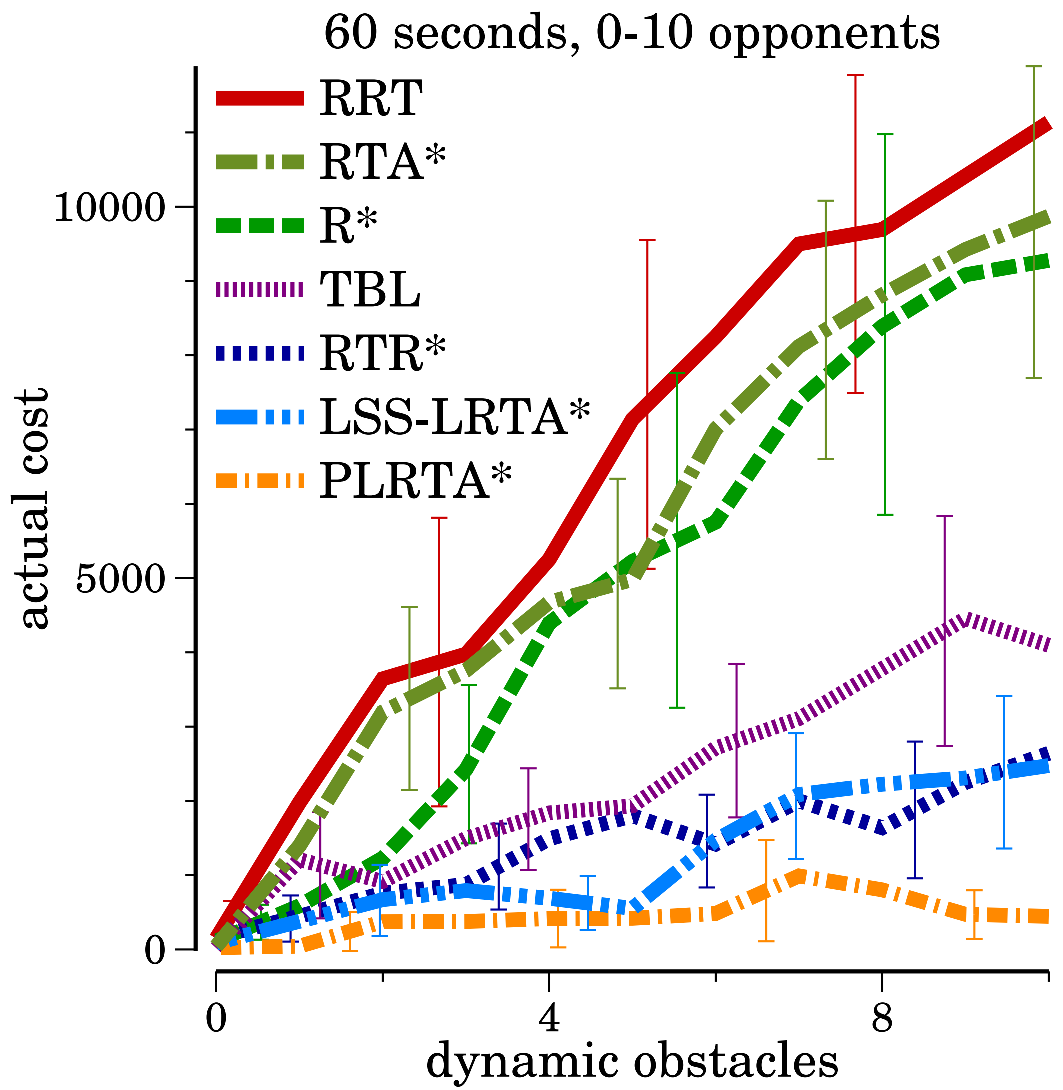

60 seconds, 8 opponents
actual cost
30000
20000
10000
RRT
R*
RTR*
TBL
PLR-TA*
LSS-LRT-A*
RTA*


60 seconds, 10 opponents
actual cost
30000
20000
10000
RRT
R*
RTR*
TBL
PLR-TA*
LSS-LRT-A*
RTA*



RTR* better than R*

Scaling



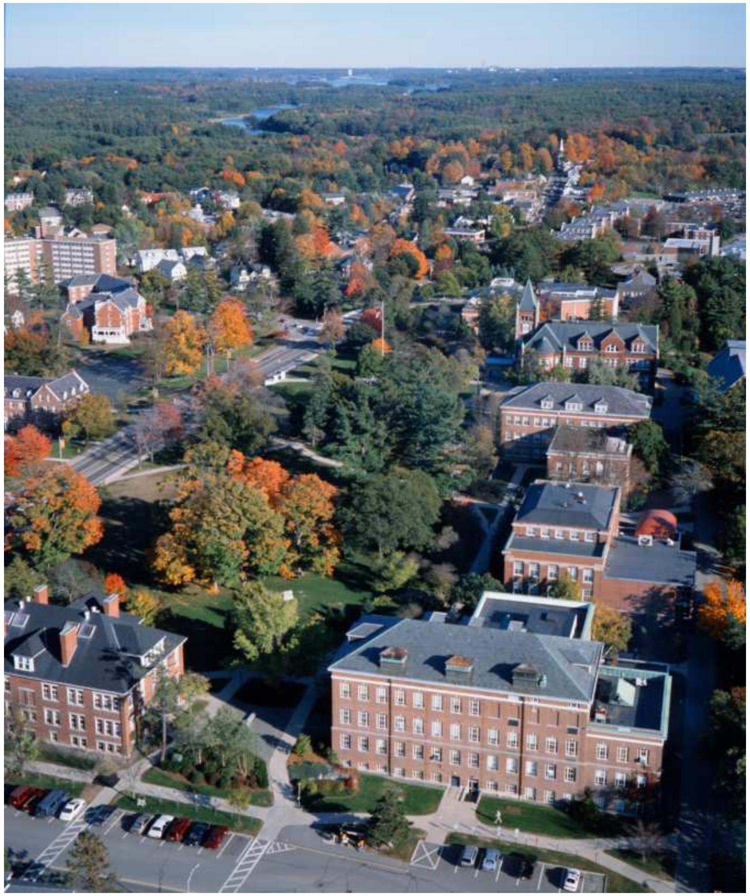
PLRTA* best

Conclusion

Conclusions

- Dynamic motion planning is a challenging domain!
- Real-time search with dynamic edge weights is important
- non-trivial to use standard algorithms
- PLRTA* is a simple hack — more work needed!

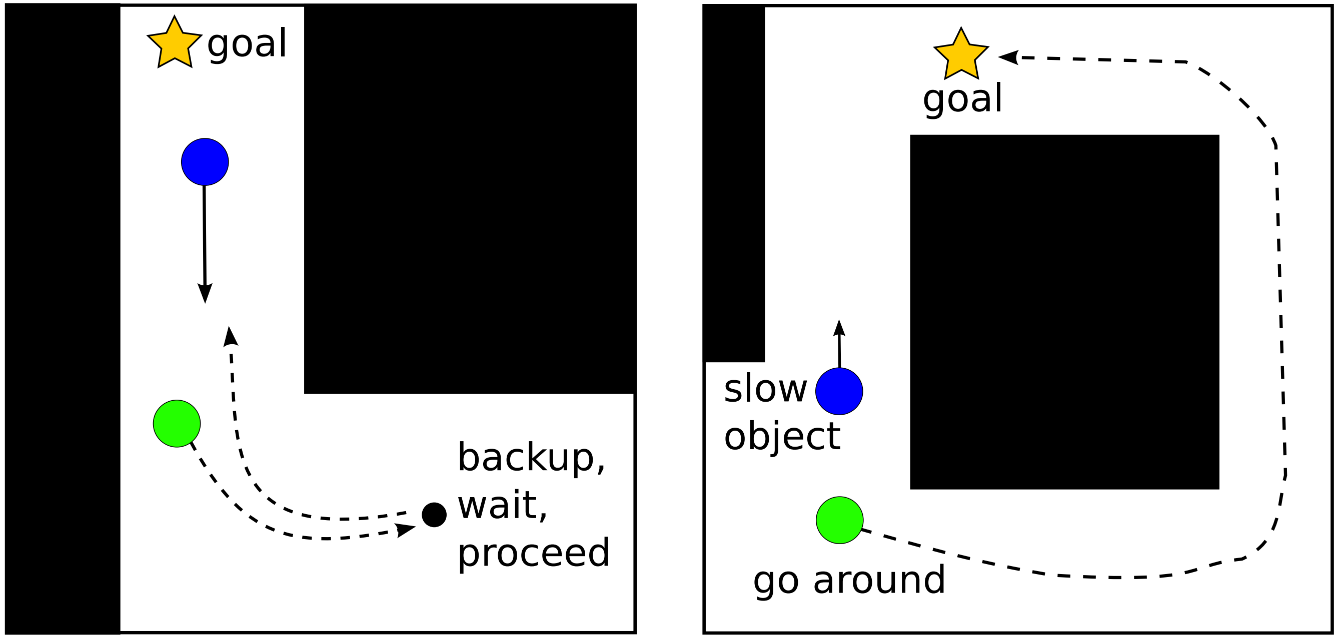
tell your students to apply to grad school in CS at UNH!



- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking, bioinformatics

Back-up Slides

Handcrafted Scenarios



6 handcrafted benchmarks, qualitative evaluation

PLRTA*: 2 bad

TBL: 3 bad (missed deadline > 10 times per instance)

others: much worse