Real Time Search in Dynamic Worlds

David M. Bond
Niels A. Widger
Wheeler Ruml

Xiaoxun Sun

University of New Hampshire

USC University of Southern California

Special thanks to Sven Koenig, Nathan Sturtevant, Brad Larsen,
NSF grant IIS-0812141 and the DARPA CSSG program

Real Time Search in Dynamic Worlds

- Real Time Search
  - Search that interleaves planning and execution of the plan
  - Hard constraint on the amount of planning that can be done each time step
- Dynamic Worlds
  - Worlds in which changes occur
  - Increases or decreases in edge cost(s)

Real Time Search in Dynamic Worlds

- Video Games
- Robotics
- Agents must return an action within a bounded time
- Doors, Bridges, Walls

Problem

- Solutions to Dynamism in Real Time Search
  - Run a real time search algorithm repetitively
    - No real time algorithm designed to work in dynamic worlds
    - Do not account for edge cost decreases

Outline

- Problem
- Previous Work
  - LSS-LRTA* – real time search
    - agent centered search
  - D* Lite – non-real time, dynamic search
- Solution - Real Time D* Lite (RTD*)
- Empirical Evaluation

Previous Work: LSS-LRTA*

- Repetitive agent centered search
- Uses A* lookahead for agent centered search
- Moves towards most promising node on the fringe of the agent centered search
- Updates cached h values of all states in local search space with Dijkstra's algorithm
- Avoids local minima through these cached values

LSS-LRTA* Video

Previous Work: D* Lite

- Dynamic non-real time search algorithm
- Search backwards from goal to agent
- When edge cost(s) change reuses previous search

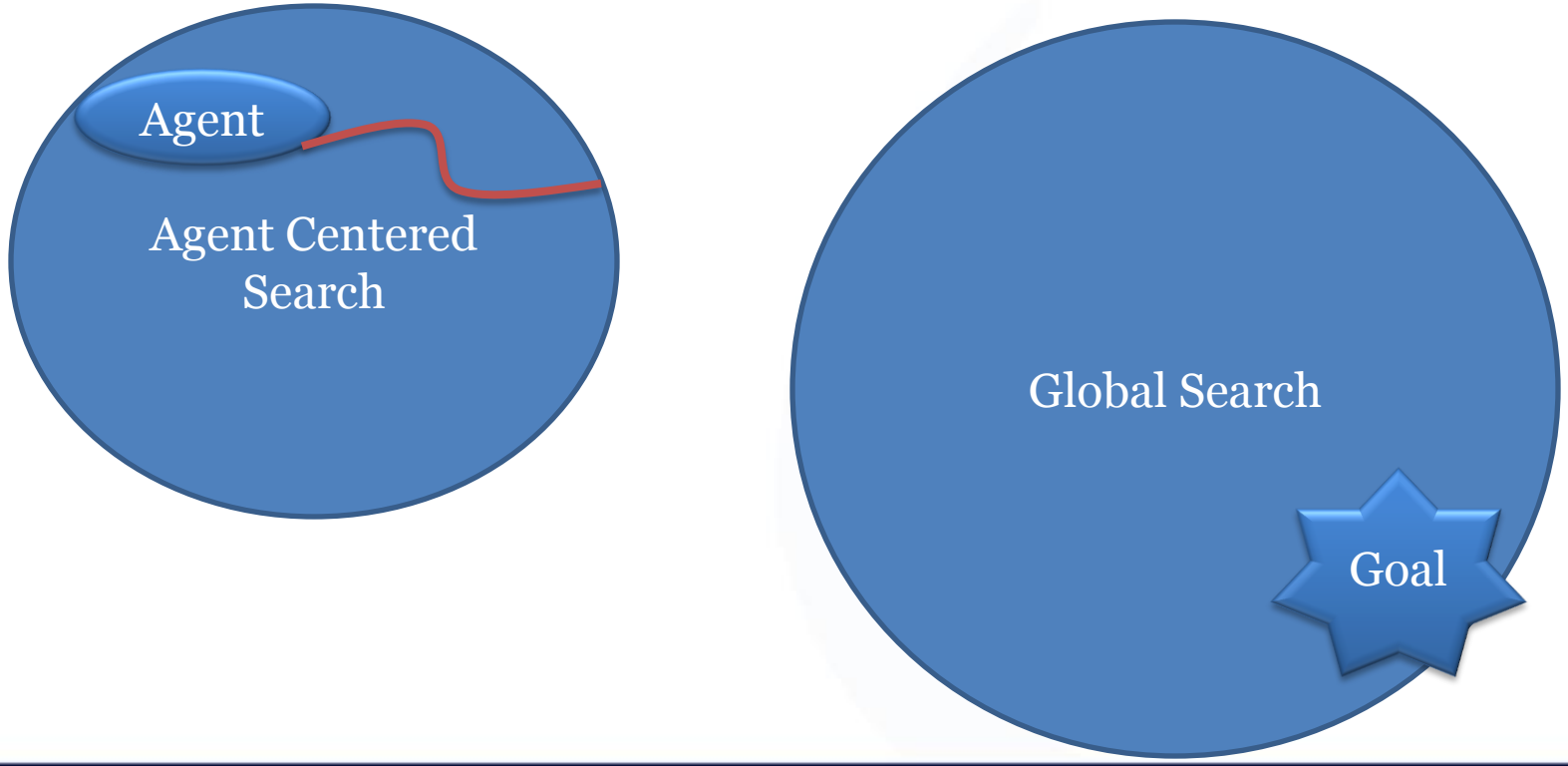
D* Lite Video

Solution: Real Time D* Lite

- Real Time
- Dynamic
- Do D* Lite
  - Stop before computation limit reached
  - Do some agent centered search
  - Resume
- Bidirectional Search
  - Use D* Lite to search from goal to agent
    - Global search
  - Use LSS-LRTA* to search from agent to goal
    - Agent centered search

Solution: Real Time D* Lite

- Combine a real time search algorithm with a dynamic search algorithm



Solution: Real Time D* Lite

- Persist global search through planning phase
- Move based on agent centered search when the global search does not have a complete path
- Move based on global search once complete path from goal to agent is found
- Computation limit divided between global and local search based on ratio parameter

Real Time D* Lite Video

Properties of Real Time D* Lite

- Complete
- Time complexity
  - Single step constant time
- Space complexity
  - Inherits from underlying algorithms

Empirical Evaluation

- Measure to evaluate a solution?
  - Time steps taken to reach goal
  - Equivalent to trajectory/path length for LSS-LRTA*, RTD*
- What algorithms used for comparison?
- LSS-LRTA*
  - Not designed to handle edge cost decreases
  - Still functions in worlds with decreases
  - May not notice new paths

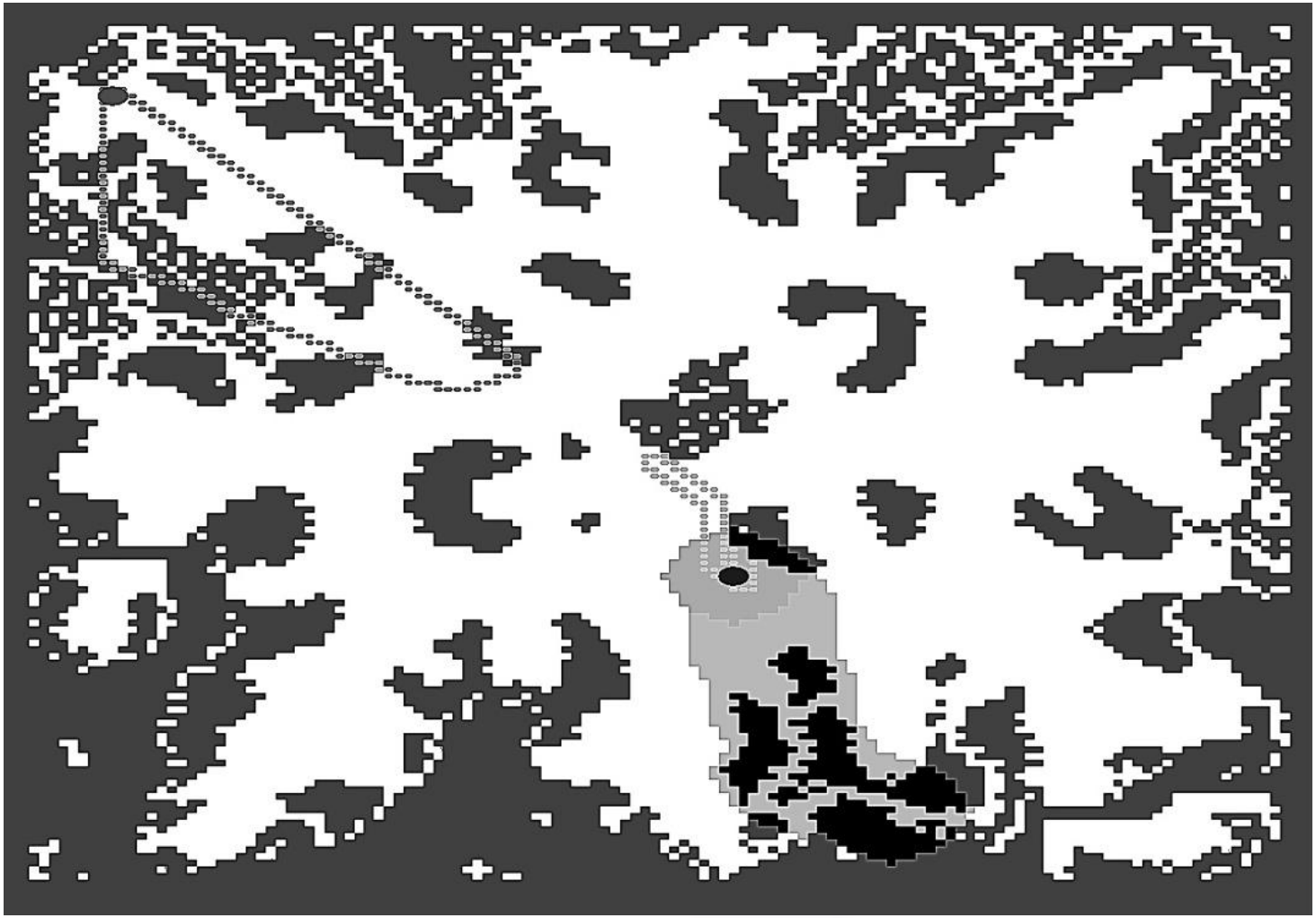
Empirical Evaluation

- Quality Bounds?
- Two variants of D* Lite
  - Non-Real Time D* Lite
    - Gives us the trajectory length of RTD* as the computation limit approaches infinity
  - Naïve real time version, RTD* NoOp
    - Agent returns do not move action after each planning phase
    - Cost of solution increases by one each time if just waiting there
    - We need to do better than this, otherwise just run D* Lite without the real time constraint

Empirical Evaluation

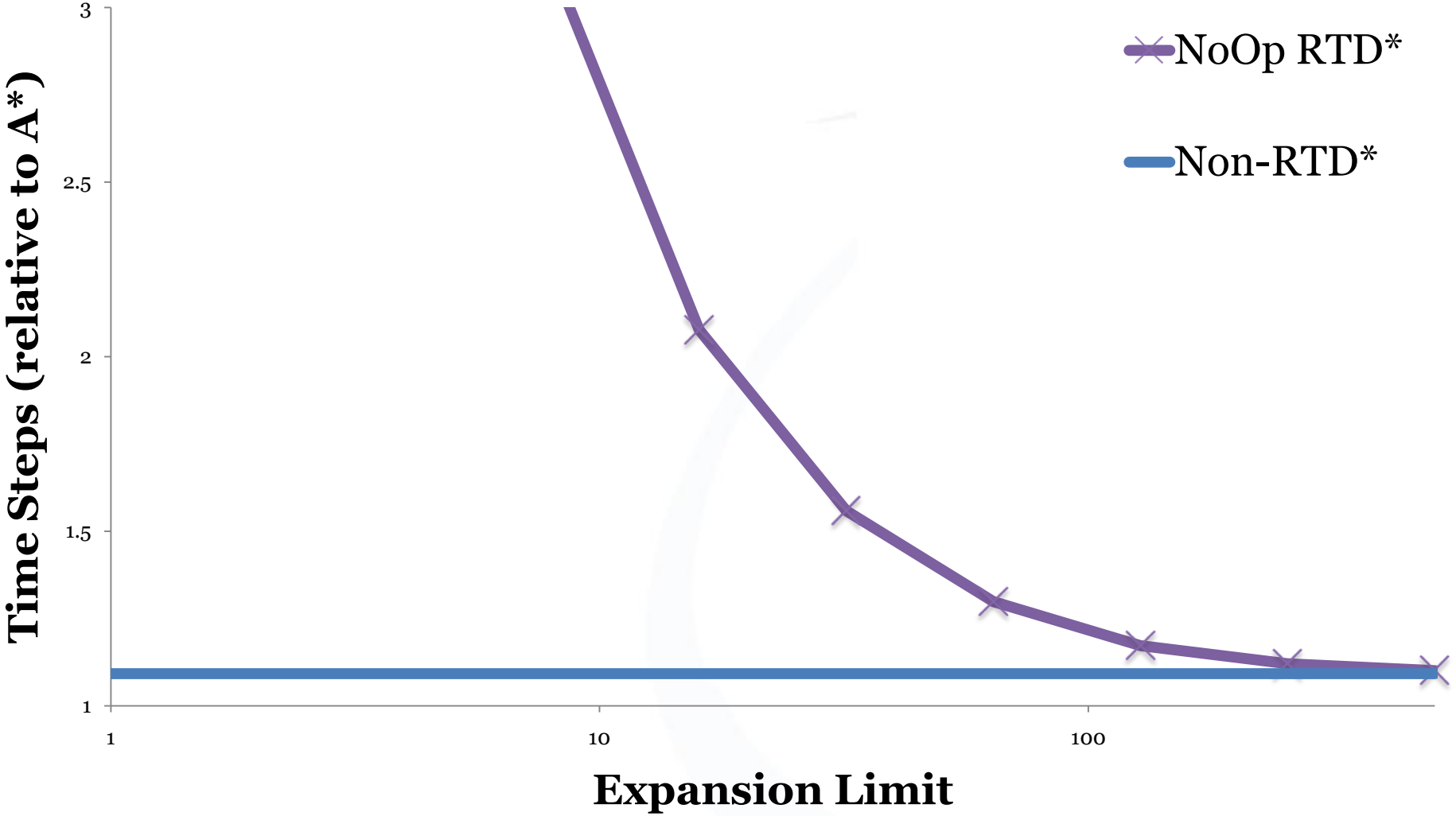
- Two domains
  - Initially unknown, static world pathfinding
    - Warcraft maps
  - Fully observable, dynamic world pathfinding
    - Rooms generated with doors opening and closing
    - Path to goal always guaranteed

Initially Unknown, Static World Pathfinding

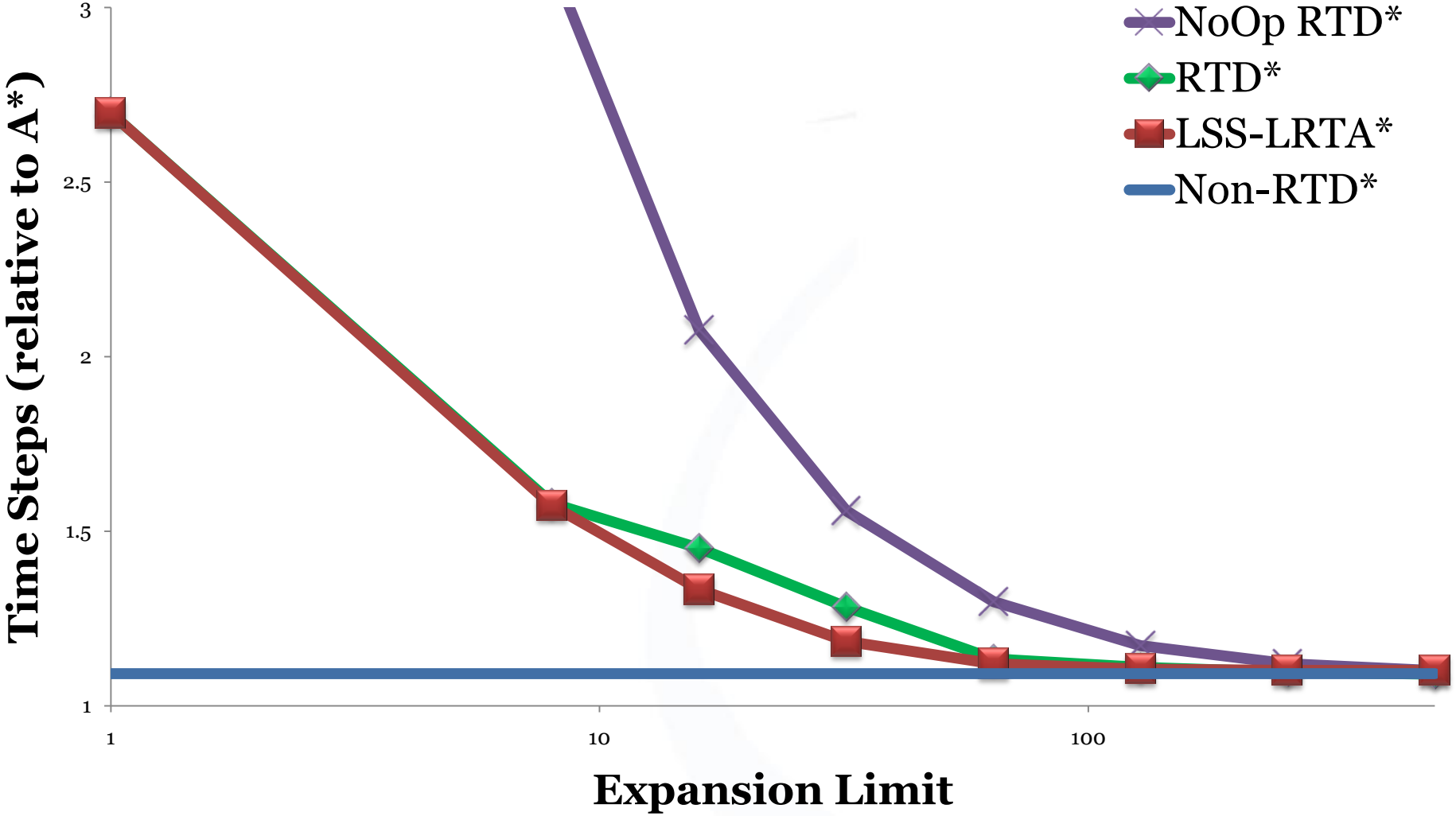


Initially Unknown, Static World Pathfinding Video

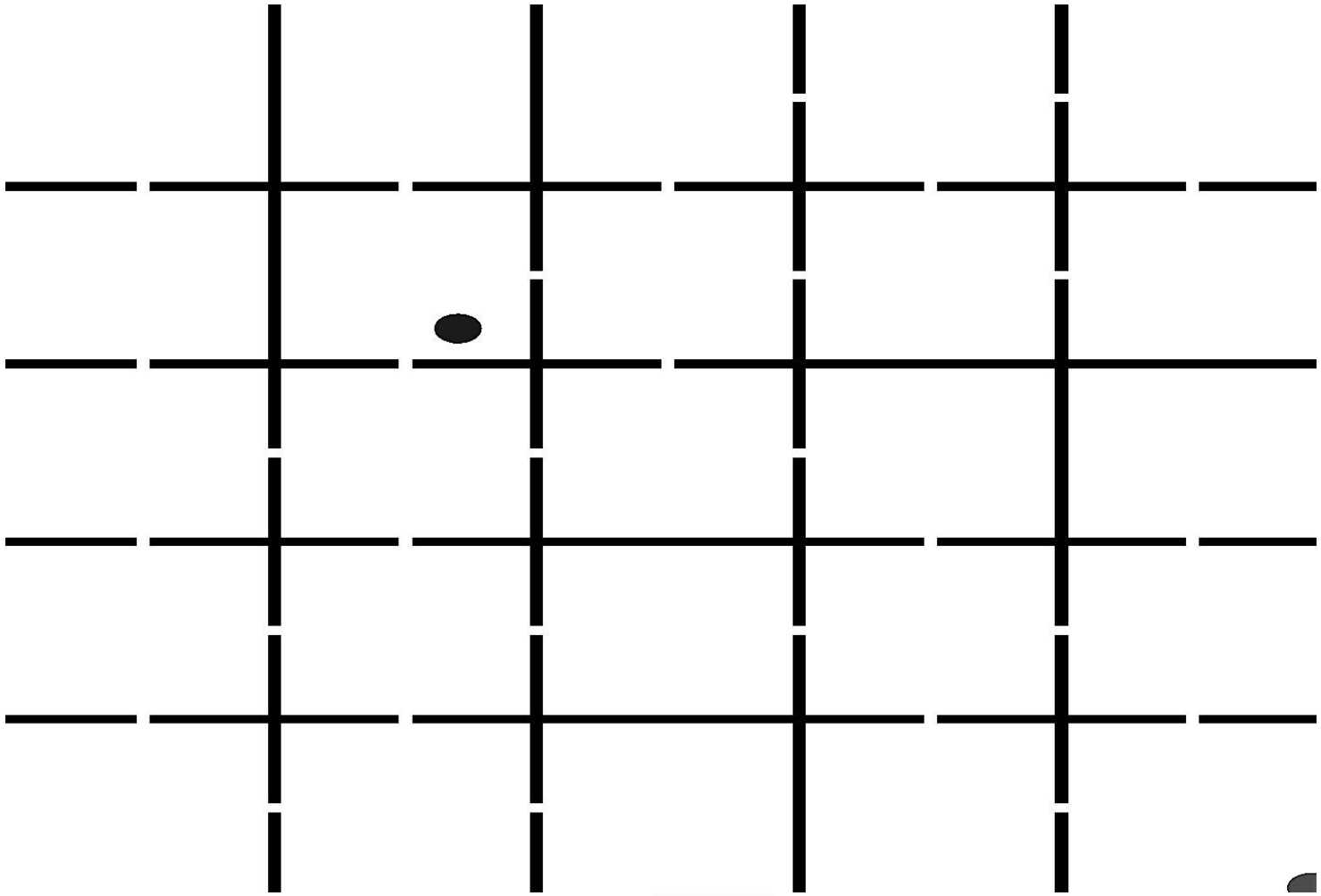
Initially Unknown, Static World Pathfinding



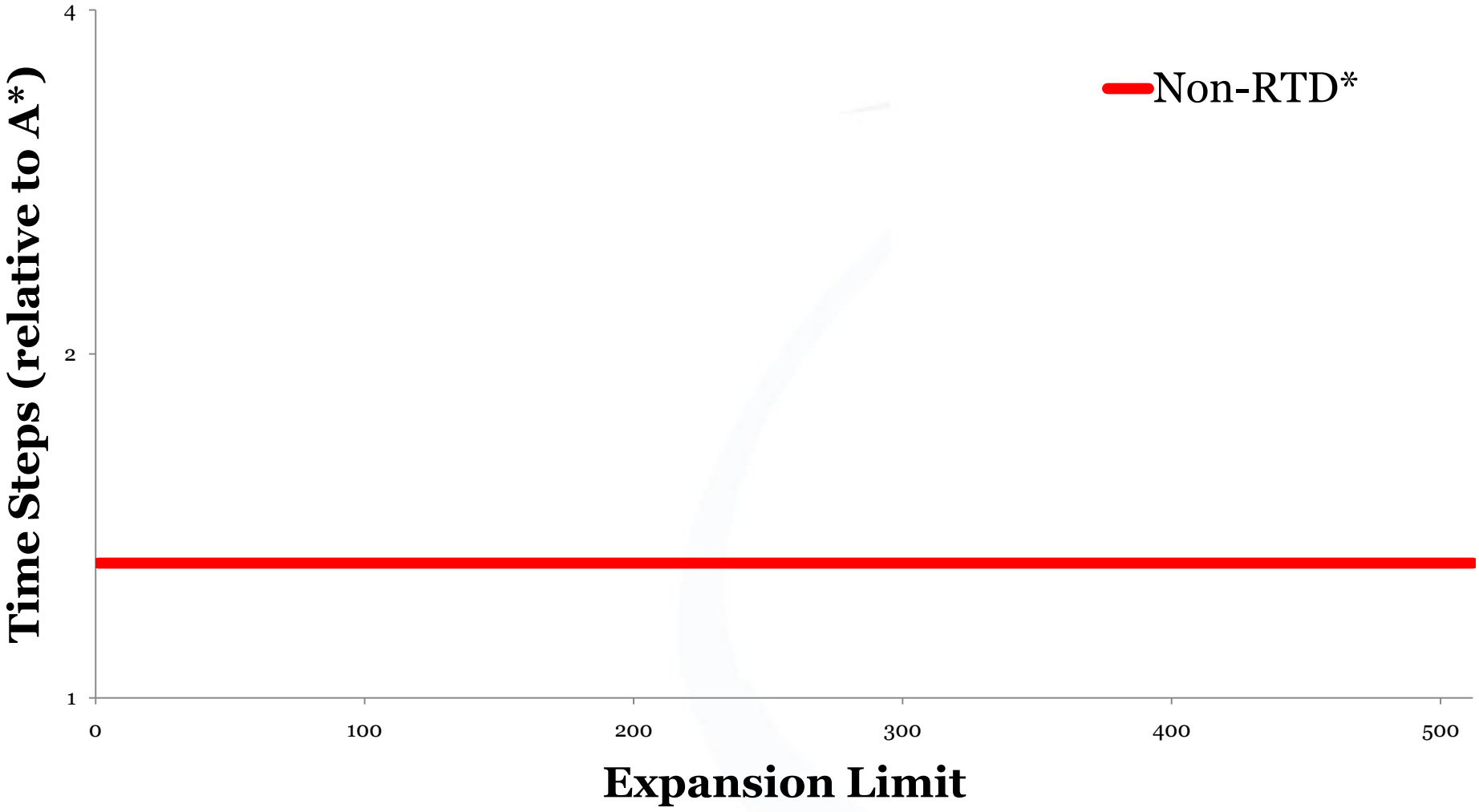
Initially Unknown, Static World Pathfinding



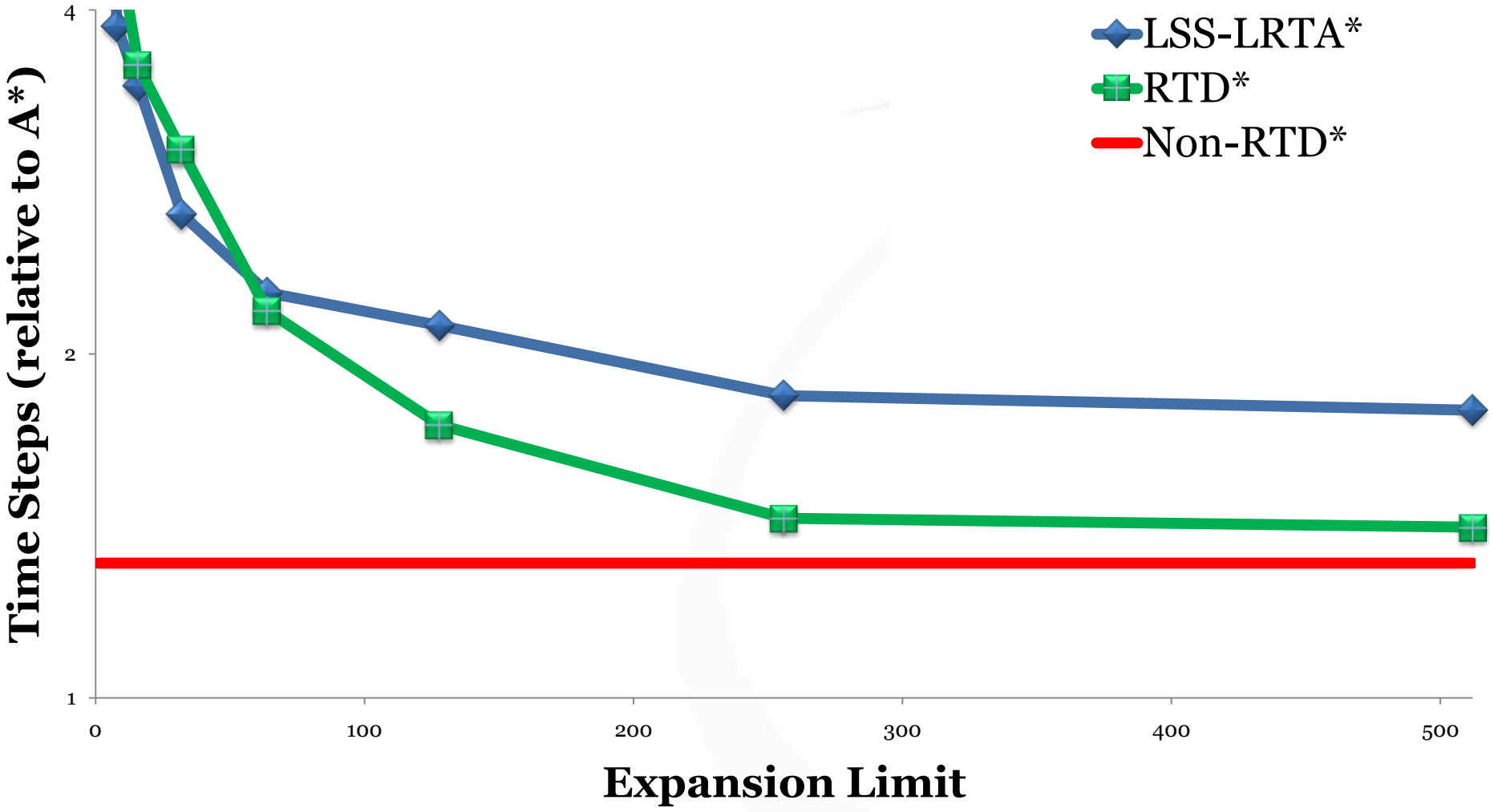
Fully Observable, Dynamic World Pathfinding



Fully Observable, Dynamic World Pathfinding



Fully Observable, Dynamic World Pathfinding



Conclusion

- The first algorithm designed for dynamic real time worlds
- Modular and expandable
- Visit http://mokon.net/mai.aspx for source code and detailed results