(1/8) Million-Module March: Scalable Locomotion for Large S-R Robots

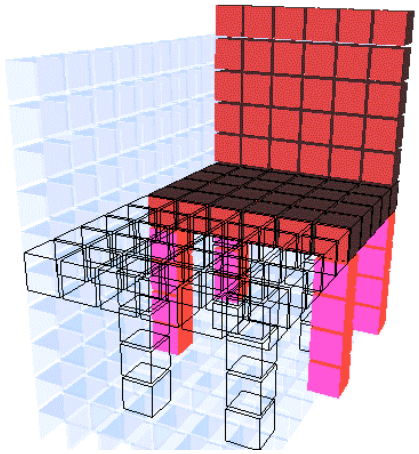
Robert Fitch
National ICT Australia

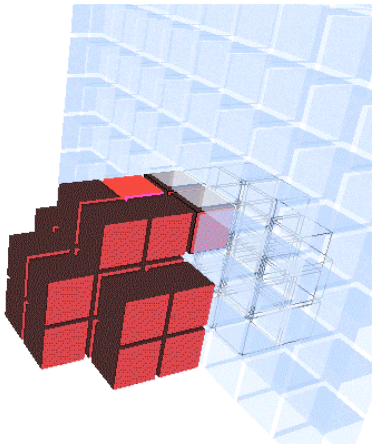
Zack Butler
Rochester Inst of Tech

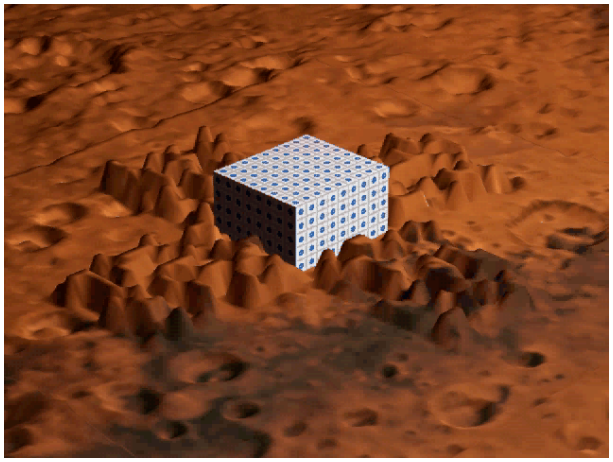
August 19, 2006

Scalability Challenges in S-R

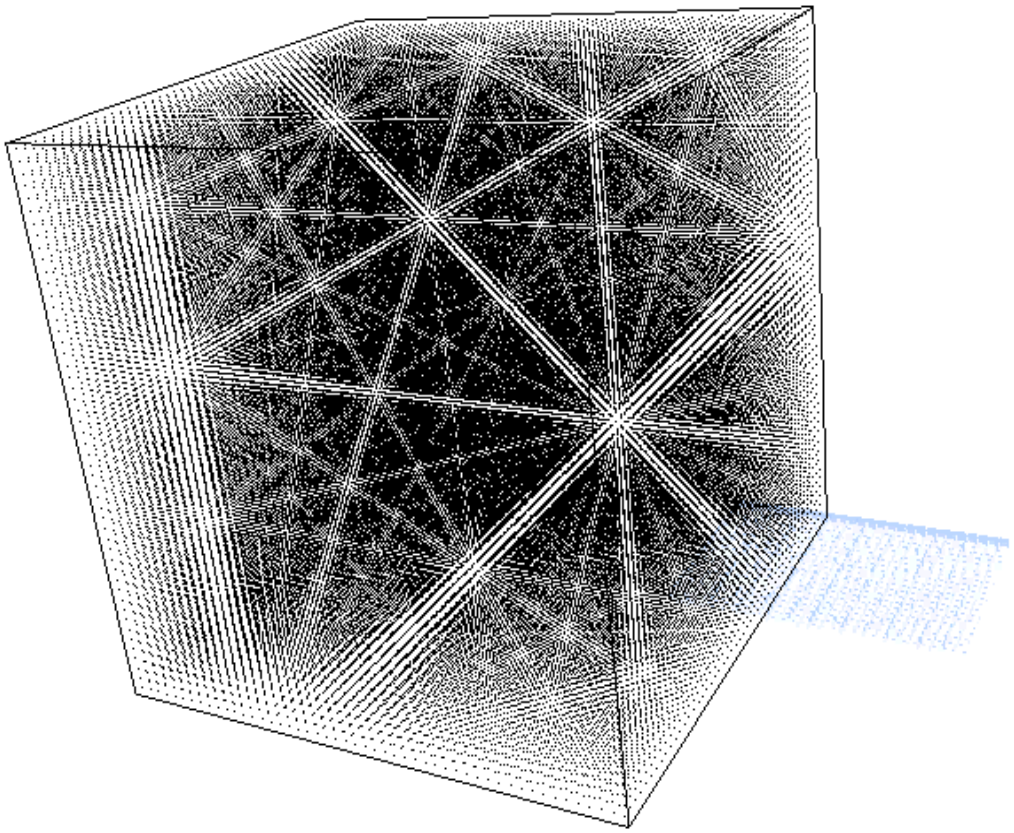
- Useful systems will probably need thousands of modules
- Each module may need to be simple
- Can't use linear space per module or linear time per action
  - Essentially precludes exact shapes?
  - Locomotion algos can use minimal space, but are restricted in





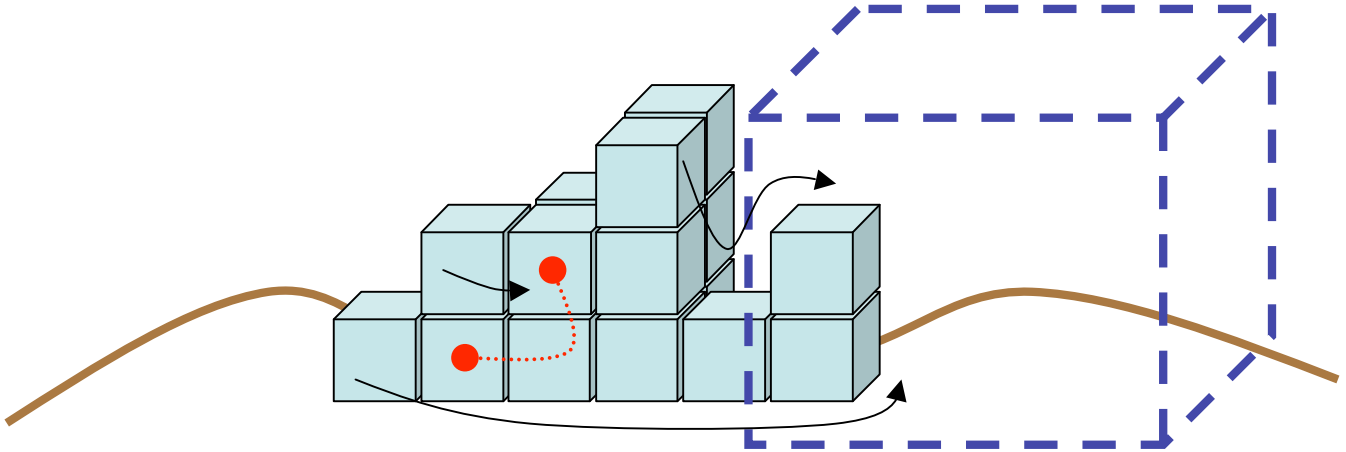


One Million (Point) Modules

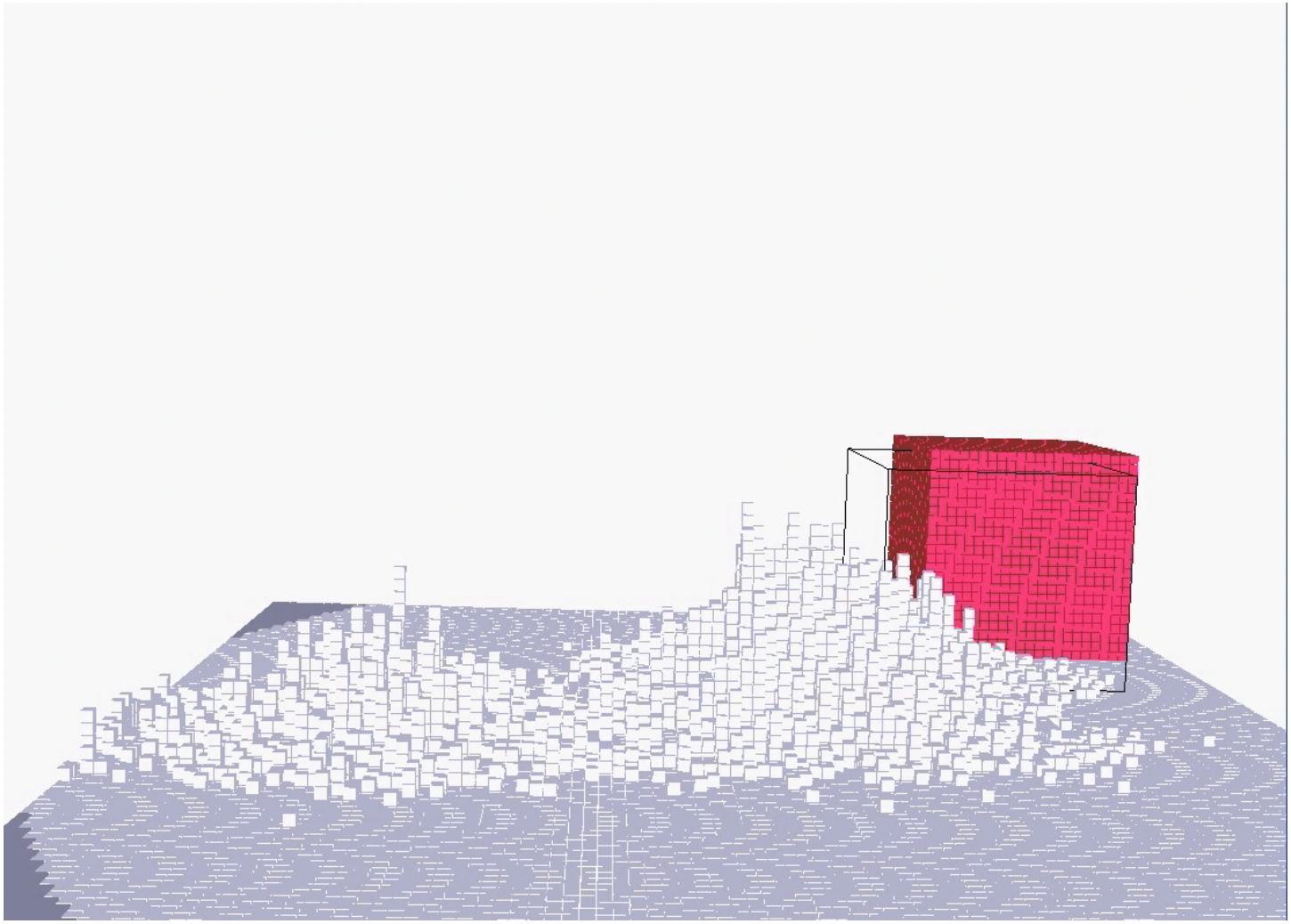


Proposed Approach

- Simplified shapes for planning
- Parallel path planning via MDP-inspired dynamic programming
- Safe parallel actuation via local connectivity checks
- Efficiency depends on shape
  - This may be true for many techniques



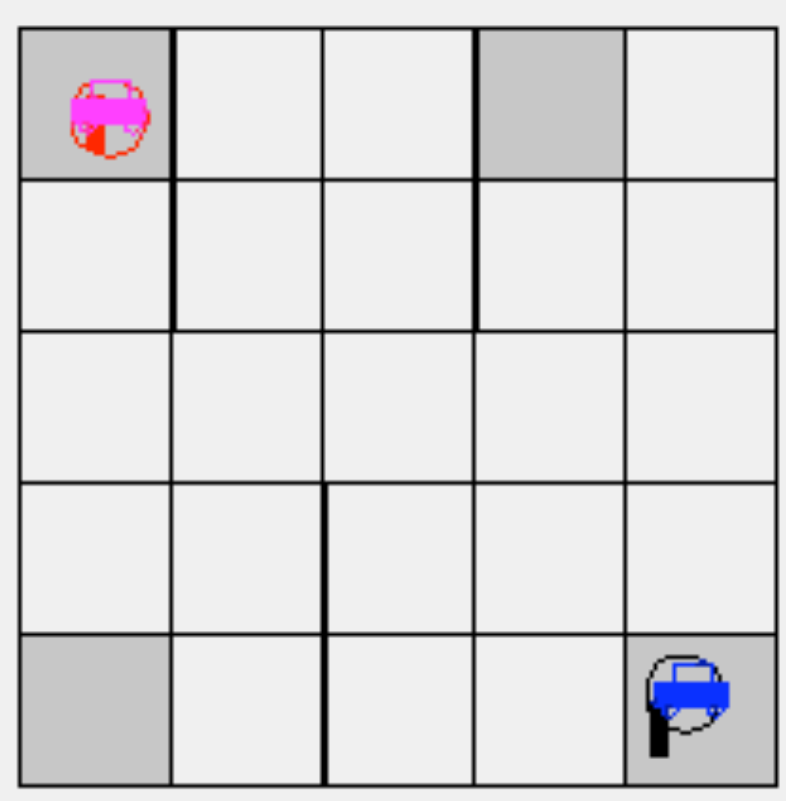
Example: Large (22^3) Robot Among Obstacles



Sublinear Planning

- Sublinear in time and space
- Bounding box to describe goal
  - Could be arbitrarily more complex
- Each module not in goal finds a path to closest goal location
  - Formulated as MDP
  - Plans are continuously updated as each module moves
  - Assuming SlidingCube abstraction, although not limited to this

MDP Formulation – GridWorld Example



MDP Formulation

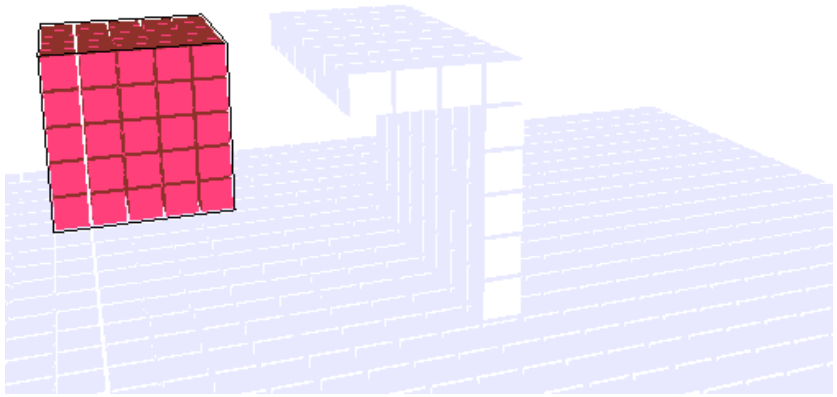
- States S: all module locations (current & potential)
- Actions A: module motions (6 faces x 4 moves + null = 25)
- Reward r = -1 (not in goal), or -k*height (in goal)
- Policy is therefore a legal motion for each location such that each module finds a goal in minimum time
- No attempt to assign goals to modules
  - Not all goal locations reachable anyway

MDP Implementation

- Each module keeps track of adjacent locations
- Assume bounding box overlaps robot
- Best policies are propagated from goal
- Module motions trigger updates

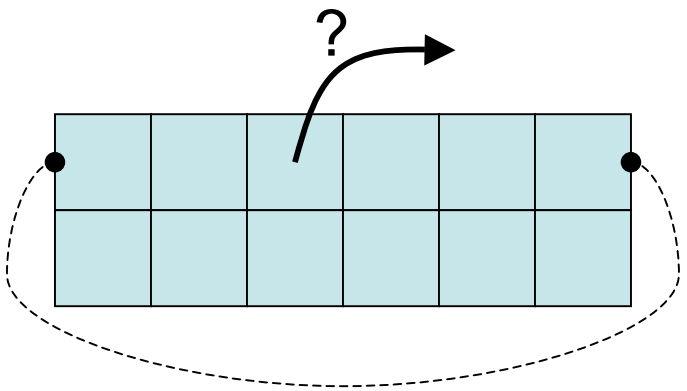
MDP Implementation

- Bounding box can overlap any obstacle
  - Even insufficient size is OK
  - Moving bounding box produces locomotion



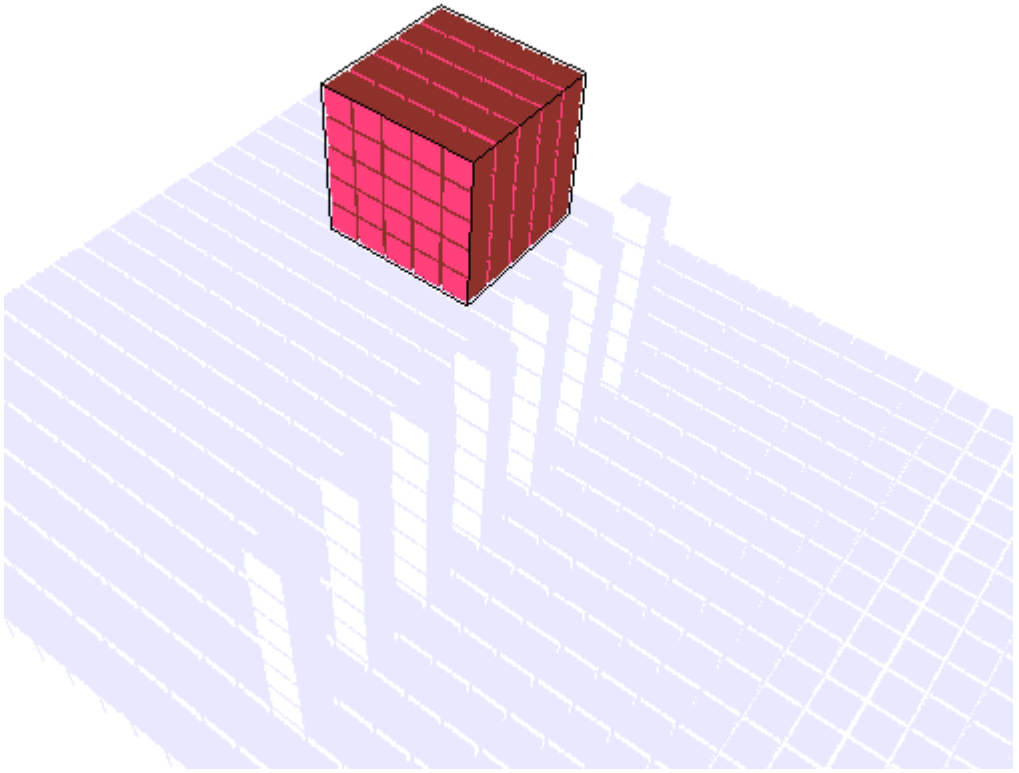
Parallel Actuation

- Must prevent global disconnection
  - Check for articulation points
  - Heuristic search ensures all neighbors remain connected after motion (find *connecting cycle*)
  - Iterative deepening limits search time
- Must avoid collisions
  - Lock modules in connecting cycle
  - Test-and-set destination position
  - Move
  - Free locked modules



Parallel Actuation

- Modules search locally & move in parallel



Efficiency Issues

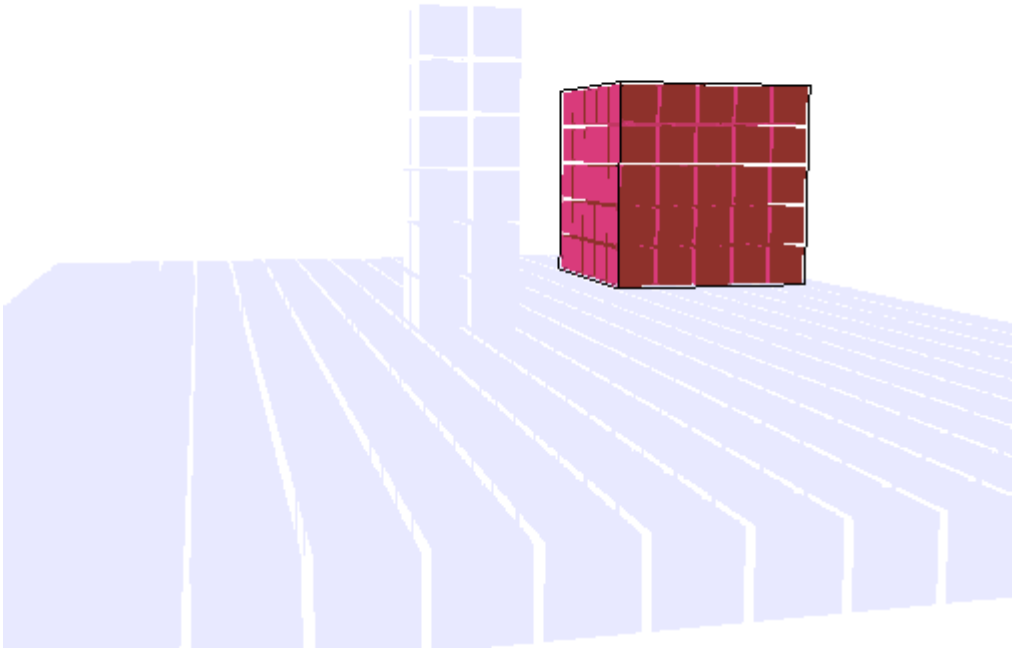
- Technique works best for blobby shapes
  - Modules multiply connected, many can move at once, bounding boxes for goals
- But gracefully degrades as well
  - Dynamic program runs in O(d) time
    - d = diameter, can be $\sqrt[3]{n}$ up to n
  - Neighbor connectivity search also will find shortest paths but as long as required

Philosophical Points

- Algorithms (for the same task) have different requirements/capabilities
- Important to consider task requirements
  - Shape fidelity, shape geometry, heterogeneity, speed of reconfiguration
- As well as (traditional) module issues
  - Memory, relative speed of communication and actuation, etc.

End

Example: Single Obstacle



Example: 1000 Modules with Obstacles