# Parallel Best-First Search: The Role of Abstraction

Ethan Burns[1], Sofia Lemons[1], Wheeler Ruml[1] and Rong Zhou[2]

1 University of New Hampshire

2 parc Palo Alto Research Center

[Many thanks to NSF grant IIS-0812141 and the DARPA CSSG program.]

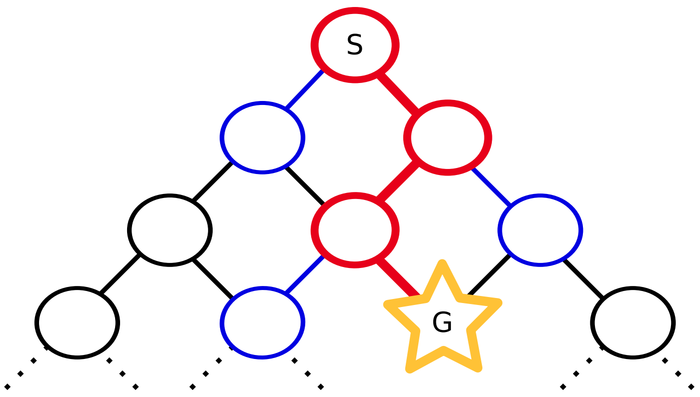
# Heuristic Search

Given:

- Initial state, Goal test, Expand function
- Cost-to-go estimate (heuristic)

Find:

- Cheapest path to a goal state




Some Properties:

- Unknown maximum depth (possibly infinite)
- Possibly duplicate states (graph, not a tree)
- Real valued edge costs

# Best-first Search



- $f(n) = g(n) + h(n)$

  $g$ is the cost accrued from initial state to $n$

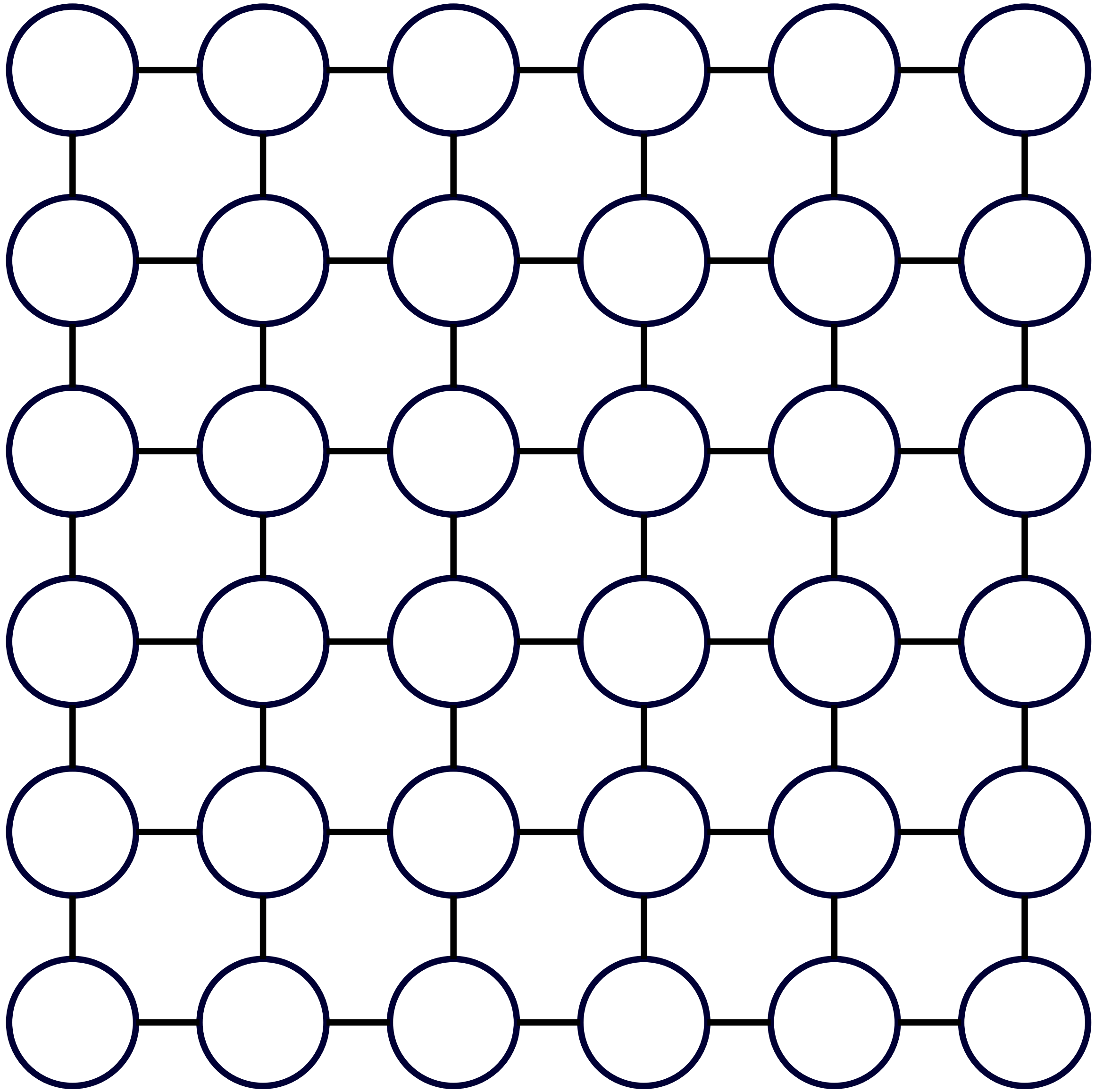
  $h$ is the estimated remaining cost to go from $n$

  $f$ is the estimated solution cost under $n$

- Search in order of lowest $f$

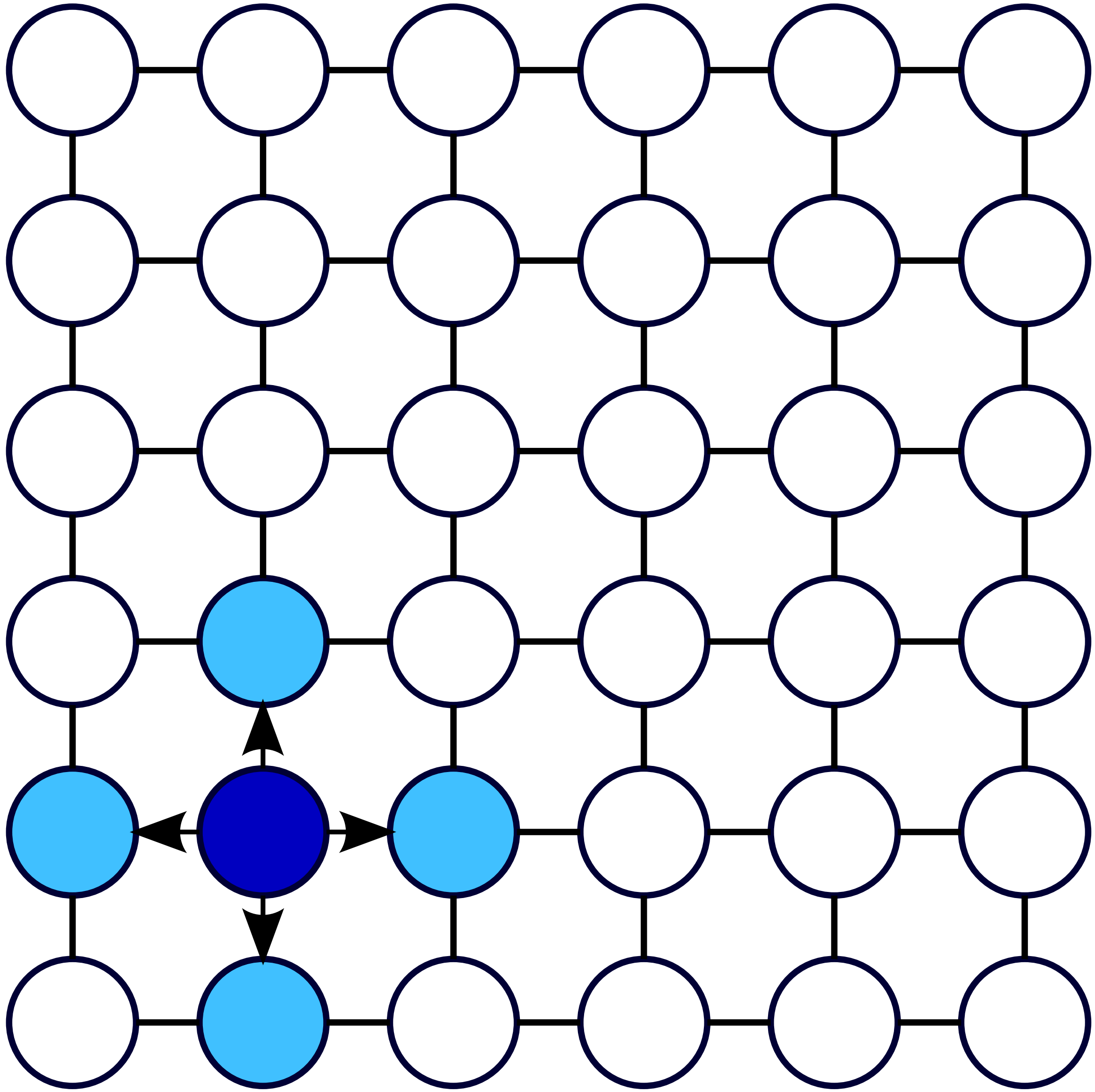
# Naive Parallel Search





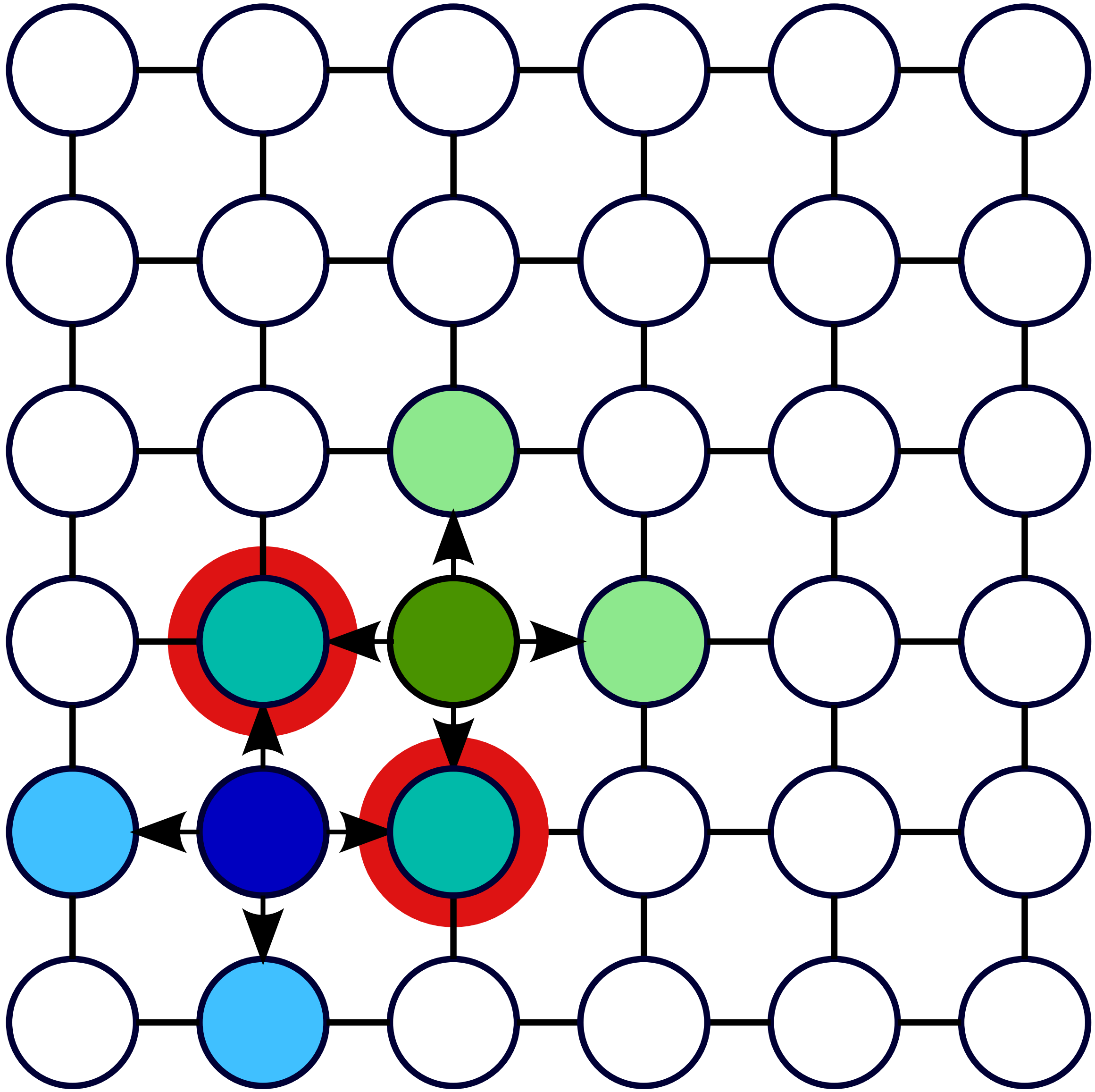
# Naive Parallel Search





# Naive Parallel Search

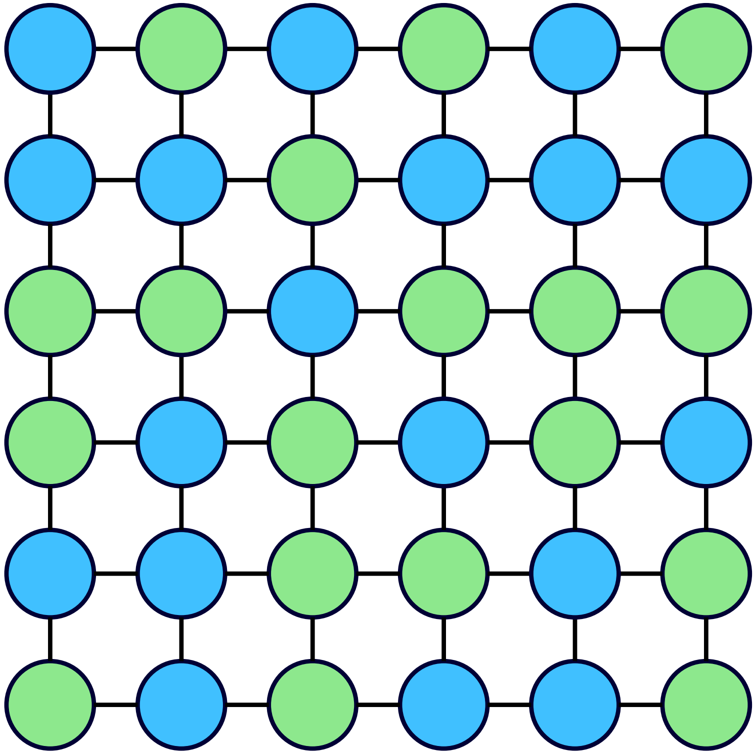




# Parallel Retracting A* (PRA*, Evett et al., 1995)



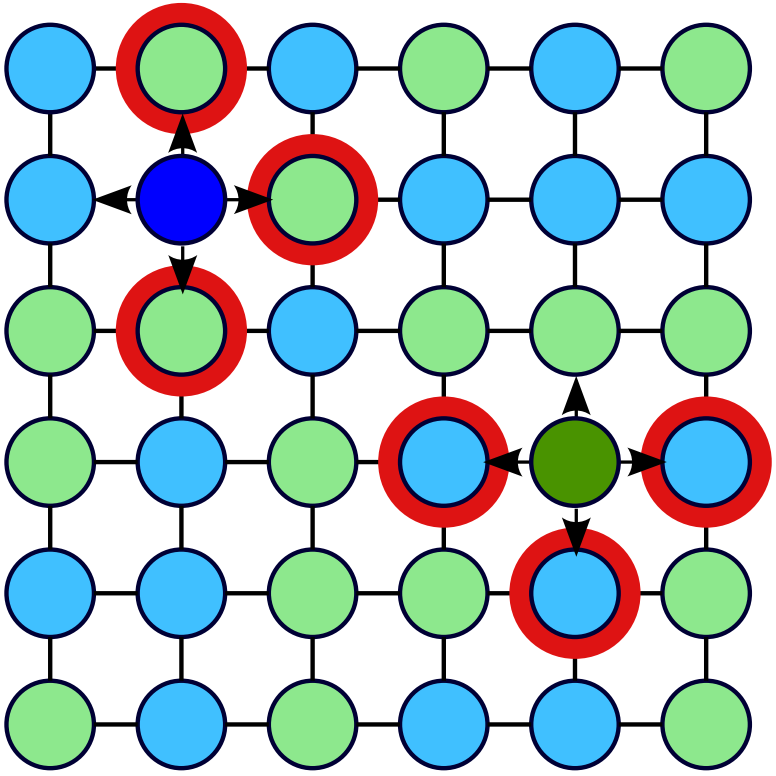
- Distribute states among threads using a hash function.
  - Each state has a home thread.
  - Duplicate detection can be performed locally at each thread.



# Parallel Retracting A* (PRA*, Evett et al., 1995)



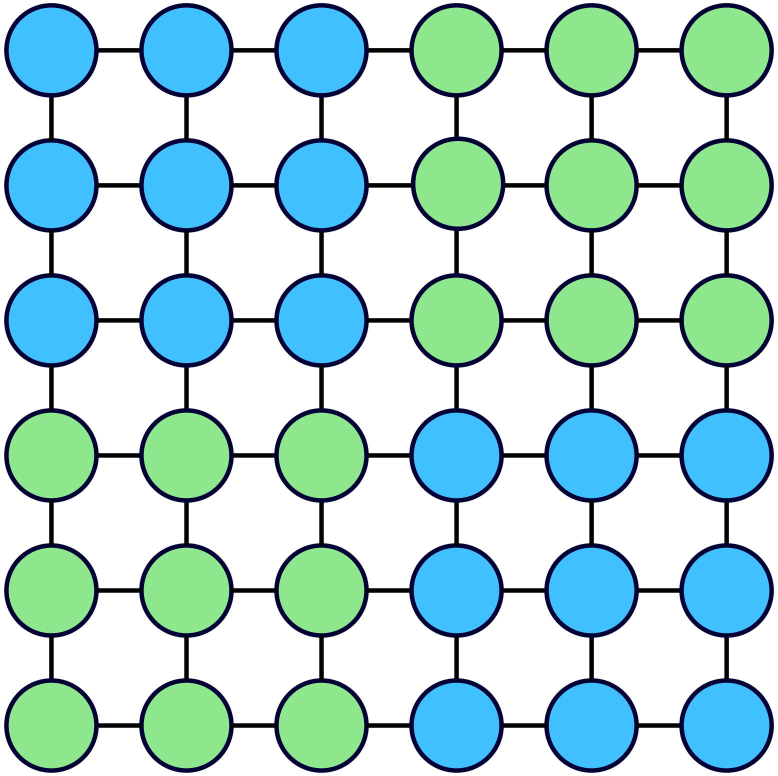
- May need to communicate states between threads at each generation.
- Non-blocking: HDA* (Kishimoto et al., best paper award ICAPS 2009)



# APRA* and AHDA*



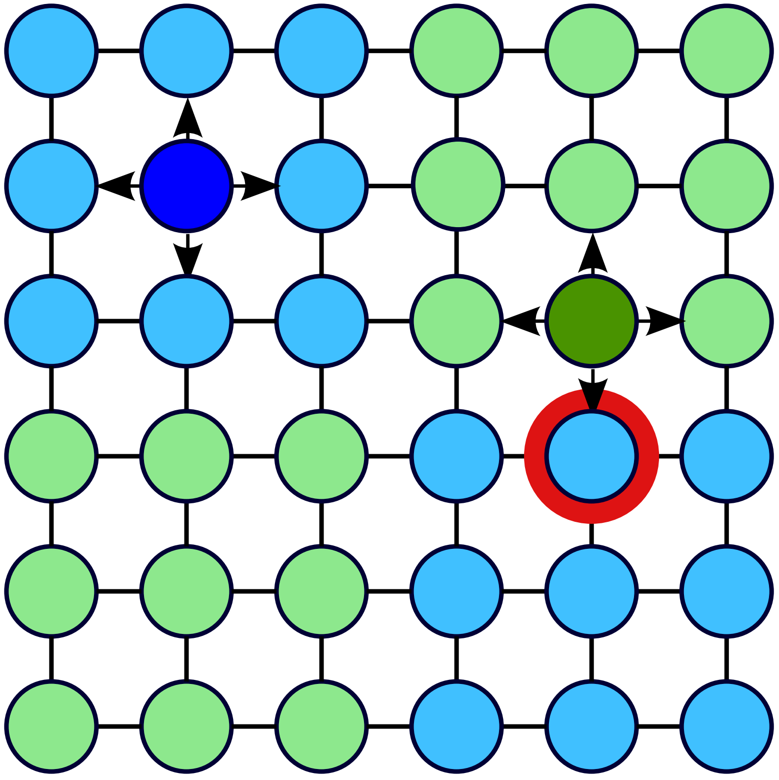
- Search space can be divided by abstraction too.
- Abstract PRA* (APRA*) and Abstract HDA* (AHDA*)



# APRA* and AHDA*



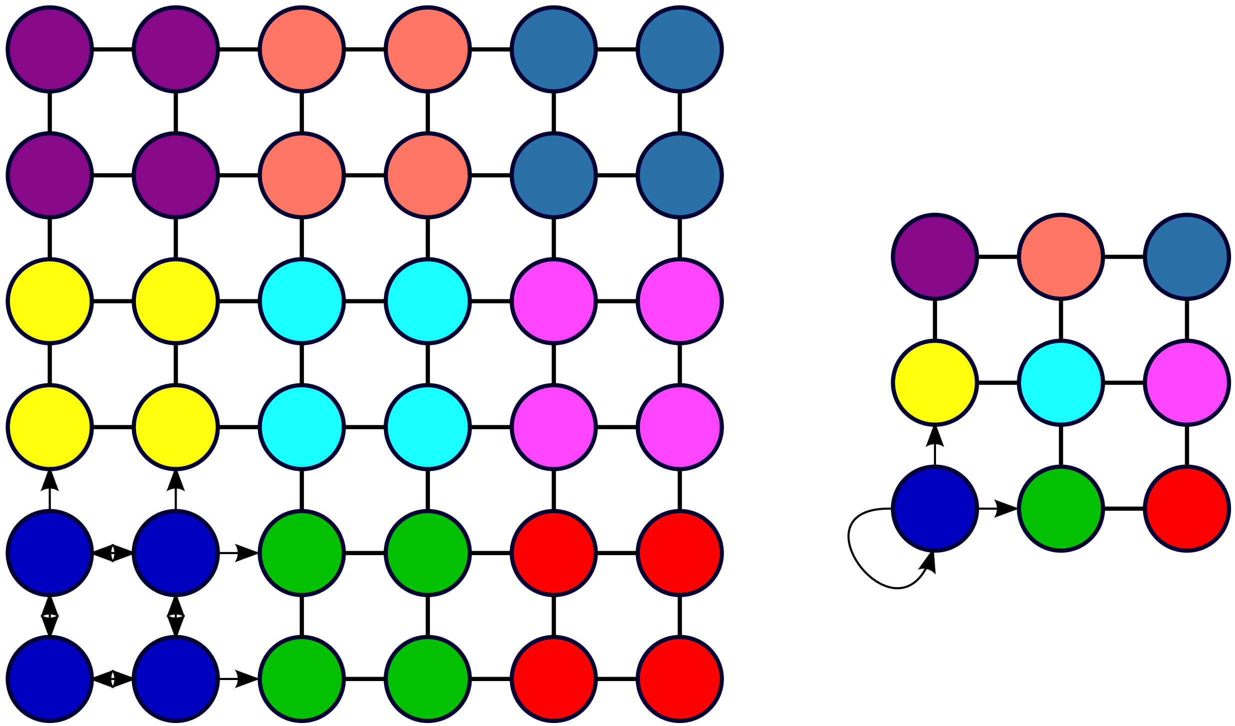
- Search space can be divided by abstraction too.
- Abstract PRA* (APRA*) and Abstract HDA* (AHDA*)



# Parallel Best $N$block First (PBNF, Burns et al., 2009)



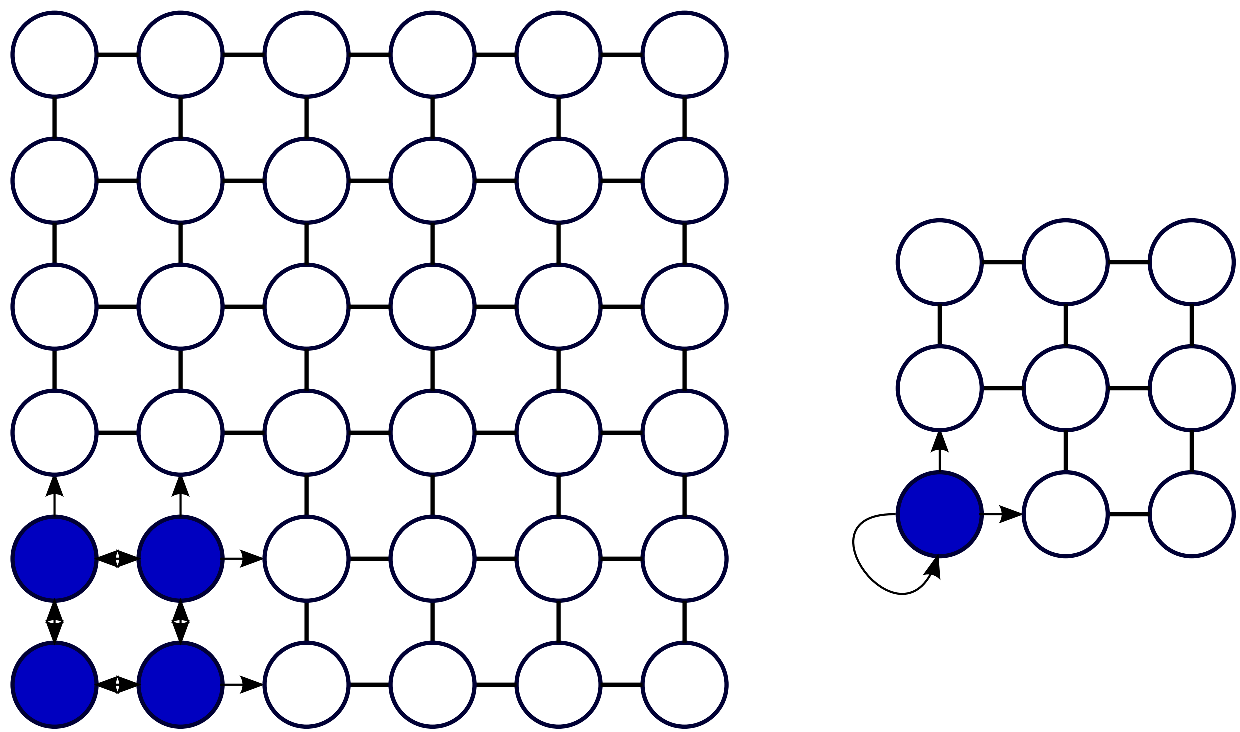
- Work is divided among threads using a special hash function based on abstraction. (Zhou and Hansen, 2007)
  - Few possible destinations for children.



# Parallel Best $N$block First (PBNF, Burns et al., 2009)



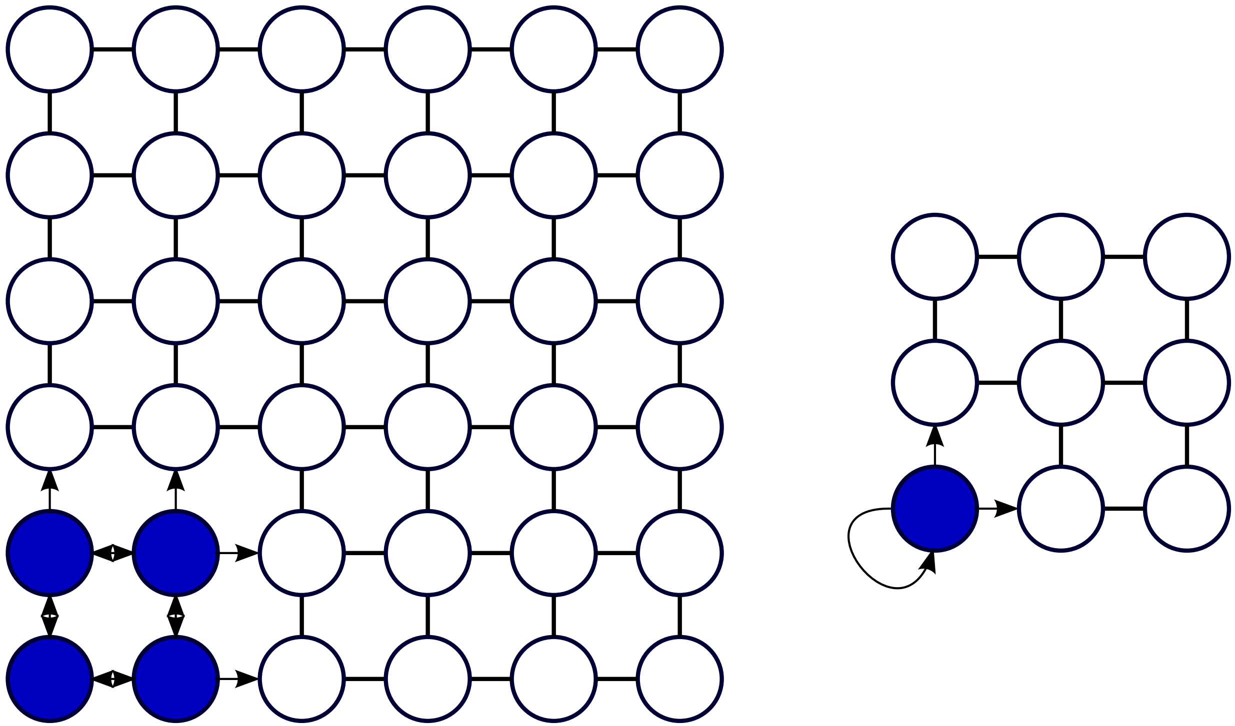
- Work is divided among threads using a special hash function based on abstraction.
  - Threads search groups of states called $n$blocks.



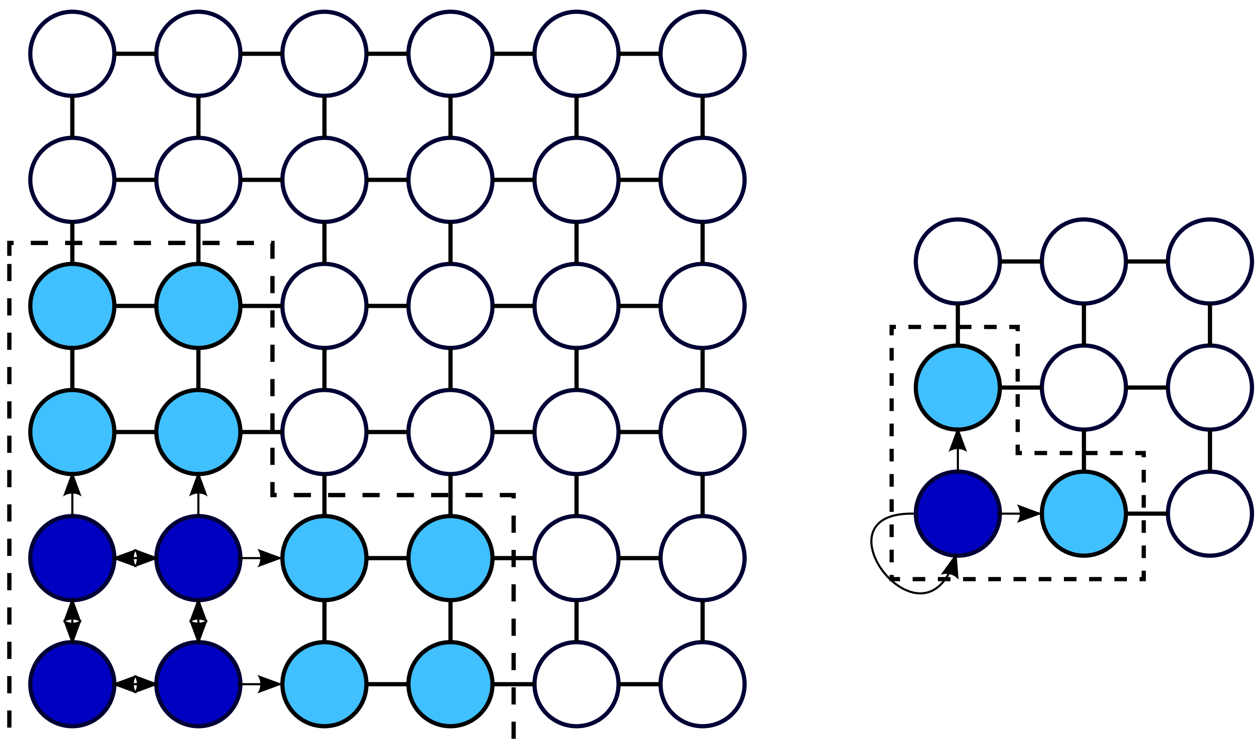
# Parallel Best $N$block First (PBNF, Burns et al., 2009)



- Work is divided among threads using a special hash function based on abstraction.
  - $n$blocks have an open and closed list.



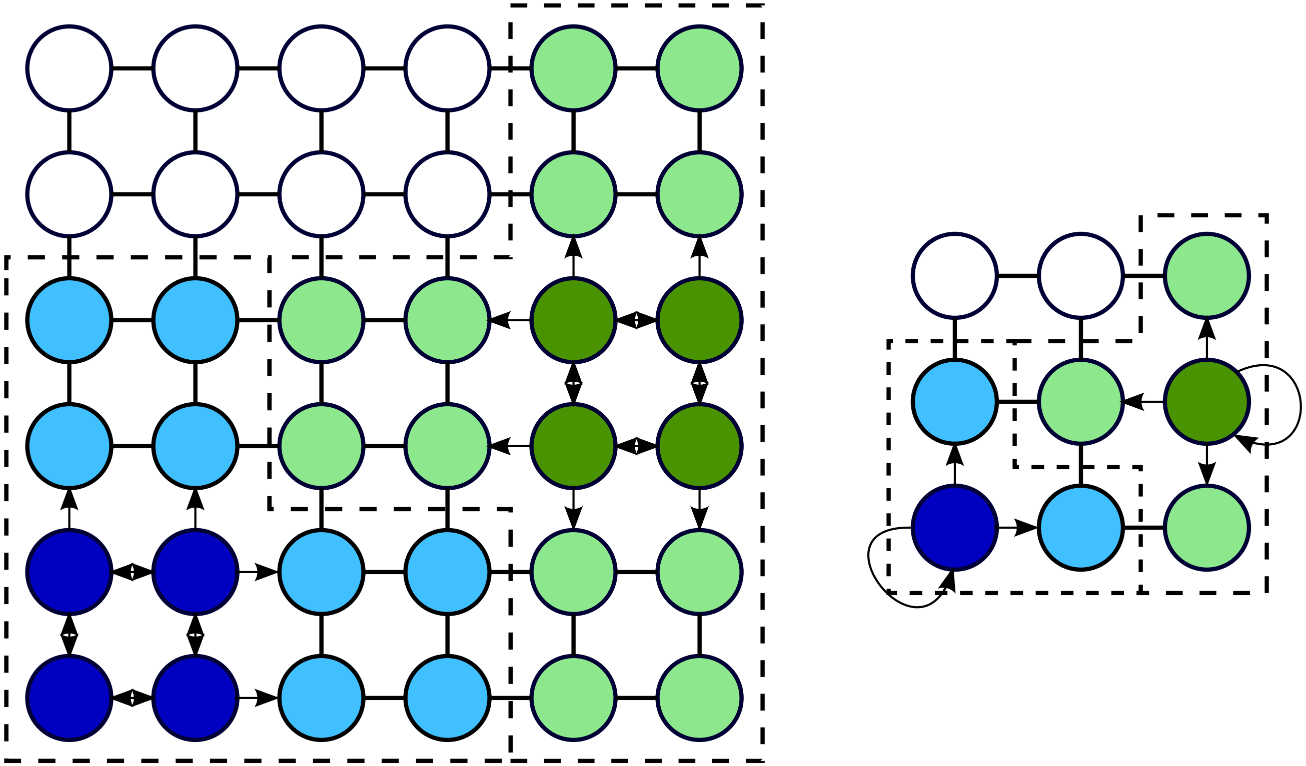
- Work is divided among threads using a special hash function based on abstraction.
  - An $n$block and its successors: *duplicate detection scope*.



# Parallel Best $N$block First (PBNF, Burns et al., 2009)



- Work is divided among threads using a special hash function based on abstraction.
  - *Disjoint* duplicate detection scopes searched in parallel.



# Parallel Best $N$Block First (PBNF, Burns et al., 2009)



1. Search disjoint $n$blocks in parallel.
   - Maintain a heap of free $n$blocks.
   - Greedily acquire best free $n$block (and its scope).
2. Each $n$block is searched in $f(n) = g(n) + h(n)$ order.
   - Switch $n$blocks when a better one becomes free.
   - Approximates best-first order.
3. Stop when the incumbent solution is optimal.
   - Prune nodes on the cost of the incumbent
   - Incumbent is optimal when all nodes are pruned.

# Outline



We have seen:

- Review of heuristic search
- Parallel search algorithms
  - PRA* (HDA*, ARPA*, AHDA*)
  - PBNF

Next:

- Parallel algorithms in optimal search.
- Parallel algorithms in bounded suboptimal search.

# Optimal Search

# Domains



Grid pathfinding:

- Navigate from start to goal in a grid maze
- Lots of ways to get to each state (lots of duplicates)

Sliding piles:

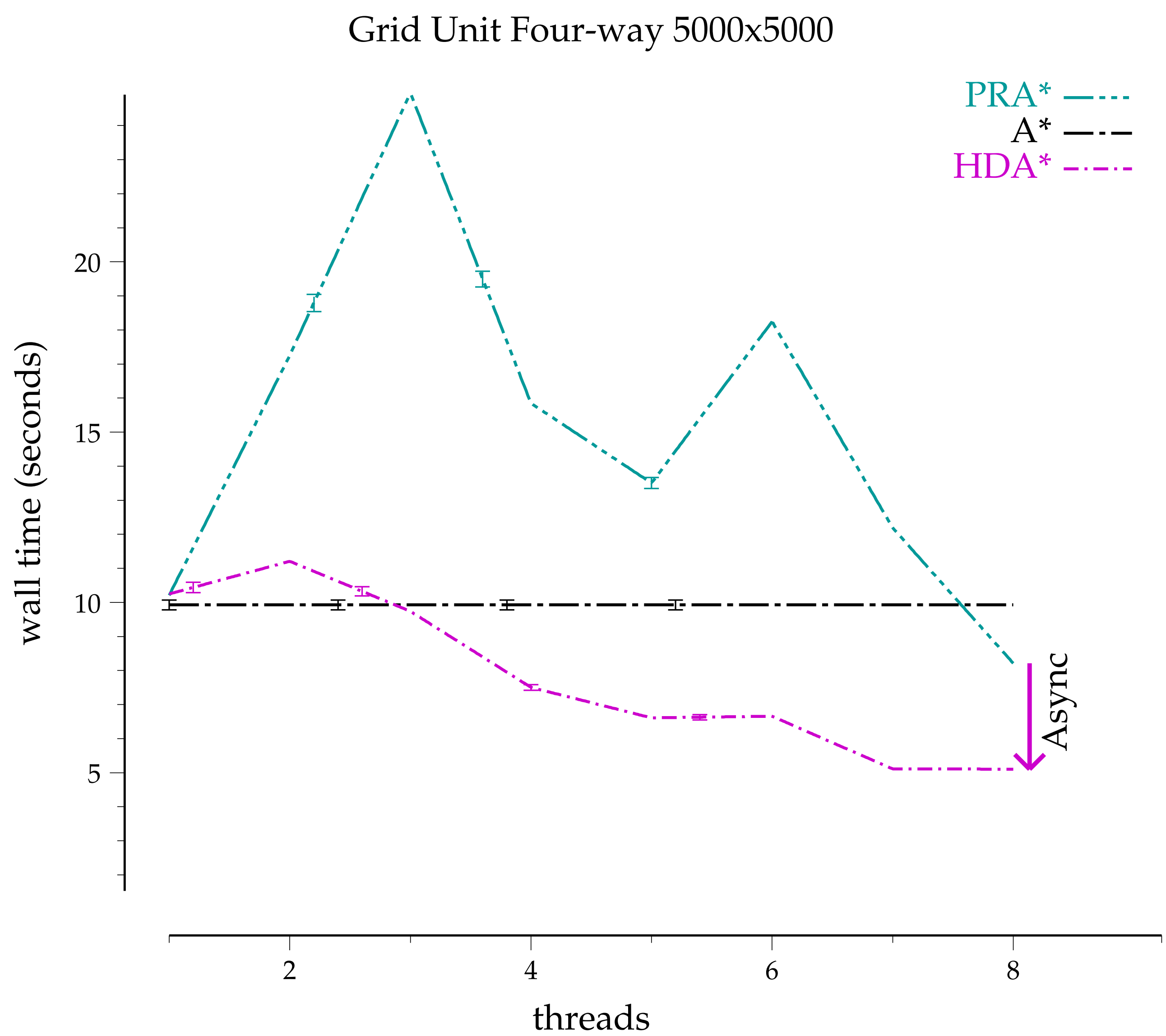
- Slide tiles around from initial to goal configuration
- Few ways to get to each state (few duplicates)

Domain independent planning:

- Find a plan in a domain given in a STRIPS-like language
- Lots of variety
- Poor quality heuristic estimate

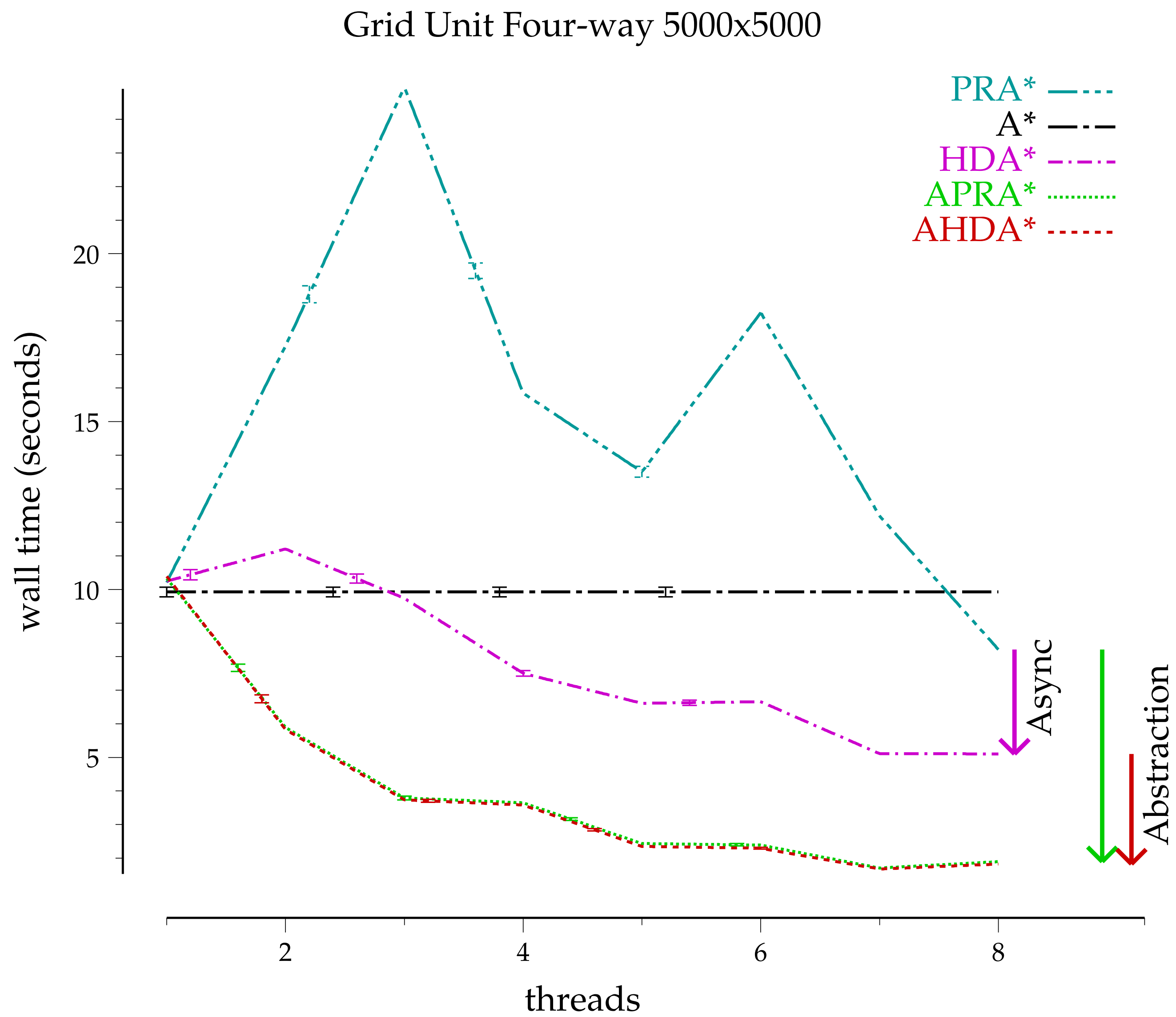
# Abstraction in PRA*





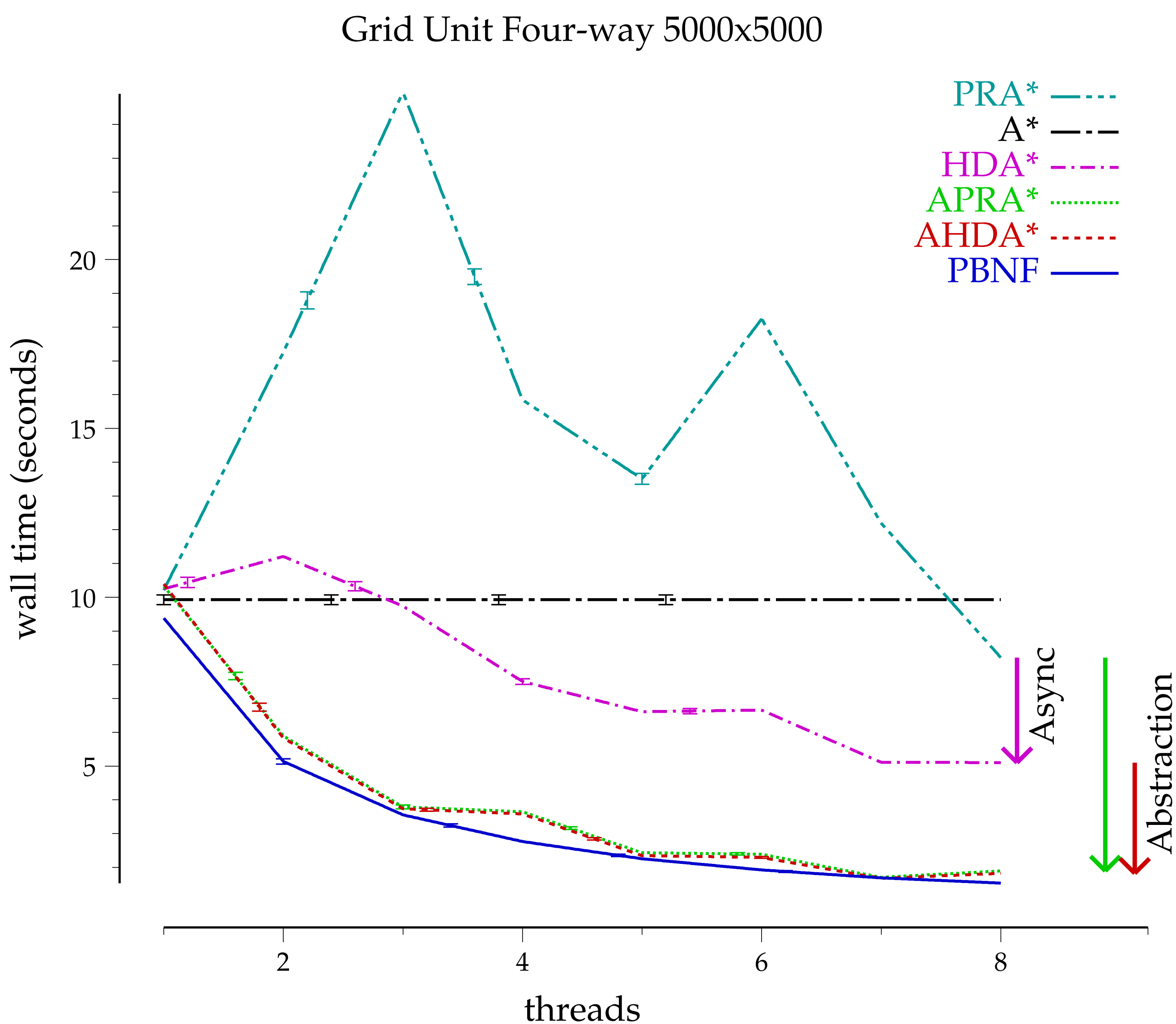
# Abstraction in PRA*





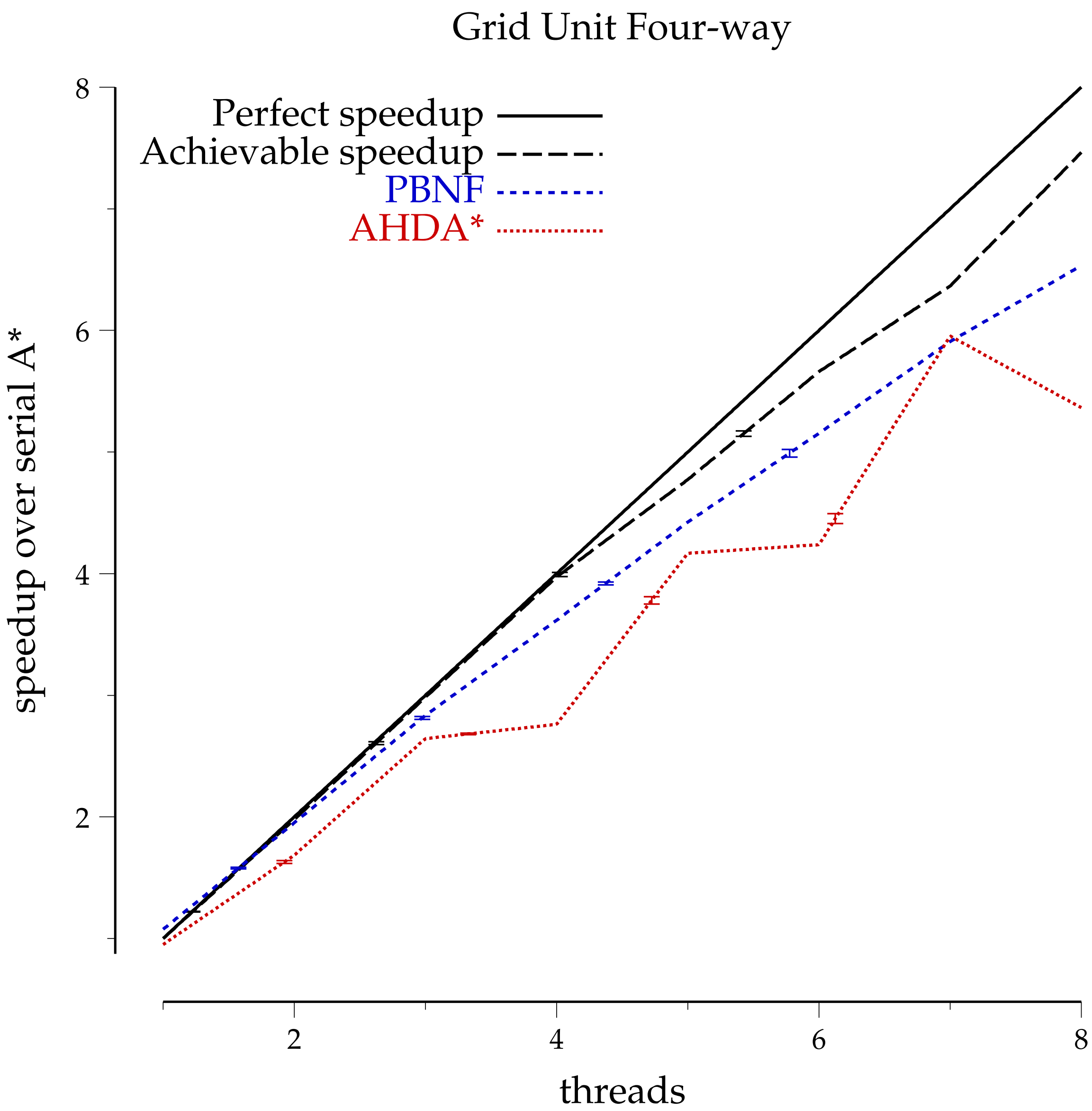
# Abstraction in PRA*





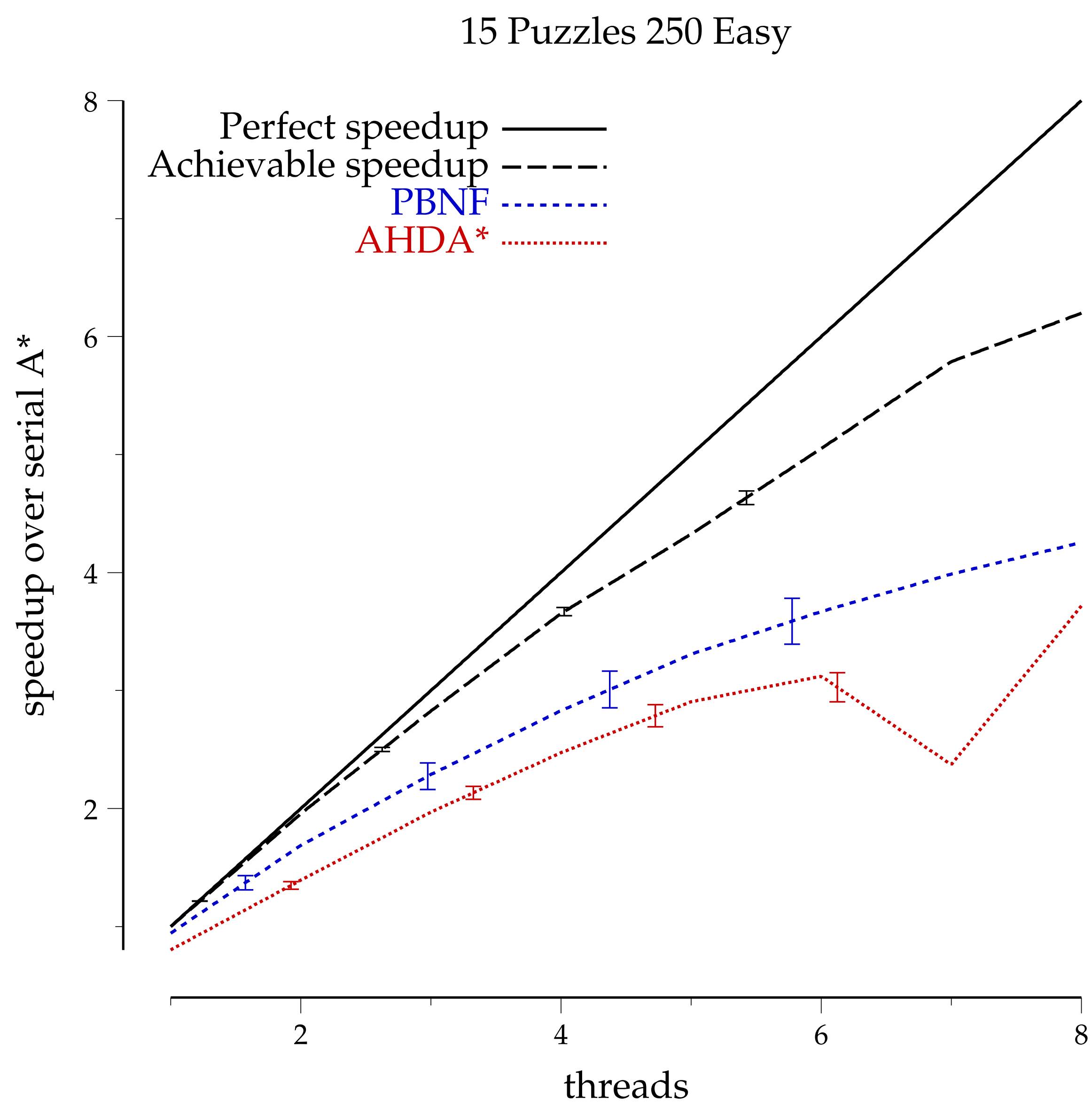
# Grid Pathfindind





# Easy 15-Puzzles





# STRIPS Planning



| | A* | AHDA* | PBNF |
|---|---|---|---|
| threads | 1 | 7 | 7 |
| logistics-6 | 2.30 | **0.40** | 0.62 |
| blocks-14 | 5.19 | 2.13 | **2.02** |
| gripper-7 | 117.78 | 12.69 | **9.21** |
| satellite-6 | 130.85 | 18.24 | **13.67** |
| elevator-12 | 335.74 | 57.10 | **27.02** |
| freecell-3 | 199.06 | **27.37** | 37.02 |
| depots-7 | - | 39.10 | **34.66** |
| driverlog-11 | - | 48.91 | **31.22** |
| gripper-8 | - | 76.34 | **51.50** |

Wall times (seconds)

# Summary of Optimal Results



- PBNF gave the best performance and scalability across all except two domains tested.
- Non-blocking communication improved the performance of PRA*, confirming results from (Kishimoto et al., 2009).
- Abstraction improved the performance of PRA* and HDA*.

# Bounded Suboptimal Search

# Bounded suboptimal



- Weighted A* searches on $f' = g + w \cdot h$
  - Finds solutions within a factor $w$ of optimal
- Converting PRA* and PBNF to bounded suboptimal (wPRA* and wPBNF)
  - Sort open lists on $f'(n) = g(n) + w \cdot h(n)$.
  - PBNF: Sort $n$block free-list on $\min_{n \in open} f'(n)$.
- Non-strict $f'$ ordering
  - Prove bound: Stop when $\min_{n \in open} w \cdot f(n) \geq g(s)$.
  - Two pruning rules: see paper.

# Four-way Grid Pathfinding 5000x5000



| | weight | threads 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| wPBNF | 1.1 | 0.84 | **1.51** | **2.23** | **2.87** | **3.41** | **4.02** | **4.55** | **5.03** |
| | 1.2 | 0.77 | 1.42 | 2.09 | 2.69 | 3.24 | 3.72 | 4.12 | 4.52 |
| | 1.4 | 0.42 | 0.92 | 1.39 | 1.83 | 2.31 | 2.51 | 2.77 | 2.98 |
| | 1.8 | 0.62 | 0.72 | 0.81 | 0.82 | 0.83 | 0.86 | 0.85 | 0.87 |
| | 3.4 | 0.71 | 0.69 | 0.69 | 0.69 | 0.67 | 0.65 | 0.64 | 0.64 |
| wAHDA* | 1.1 | **0.87** | 1.41 | 2.04 | 1.82 | 2.74 | 3.40 | 4.09 | 3.57 |
| | 1.2 | 0.79 | 1.22 | 1.82 | 1.75 | 3.28 | 3.29 | 3.96 | 3.48 |
| | 1.4 | 0.31 | 0.69 | 1.51 | 1.55 | 2.62 | 2.47 | 3.05 | 2.68 |
| | 1.8 | 0.55 | 0.74 | 0.94 | 0.69 | 0.83 | 0.81 | 0.74 | 0.64 |
| | 3.4 | 0.71 | 0.69 | 0.73 | 0.51 | 0.59 | 0.59 | 0.56 | 0.48 |

Speedup over serial wA

- wPBNF gave the best performance at all but 1 thread.
- Lower weight gives more speedup.

# Korf's 100 15-Puzzles



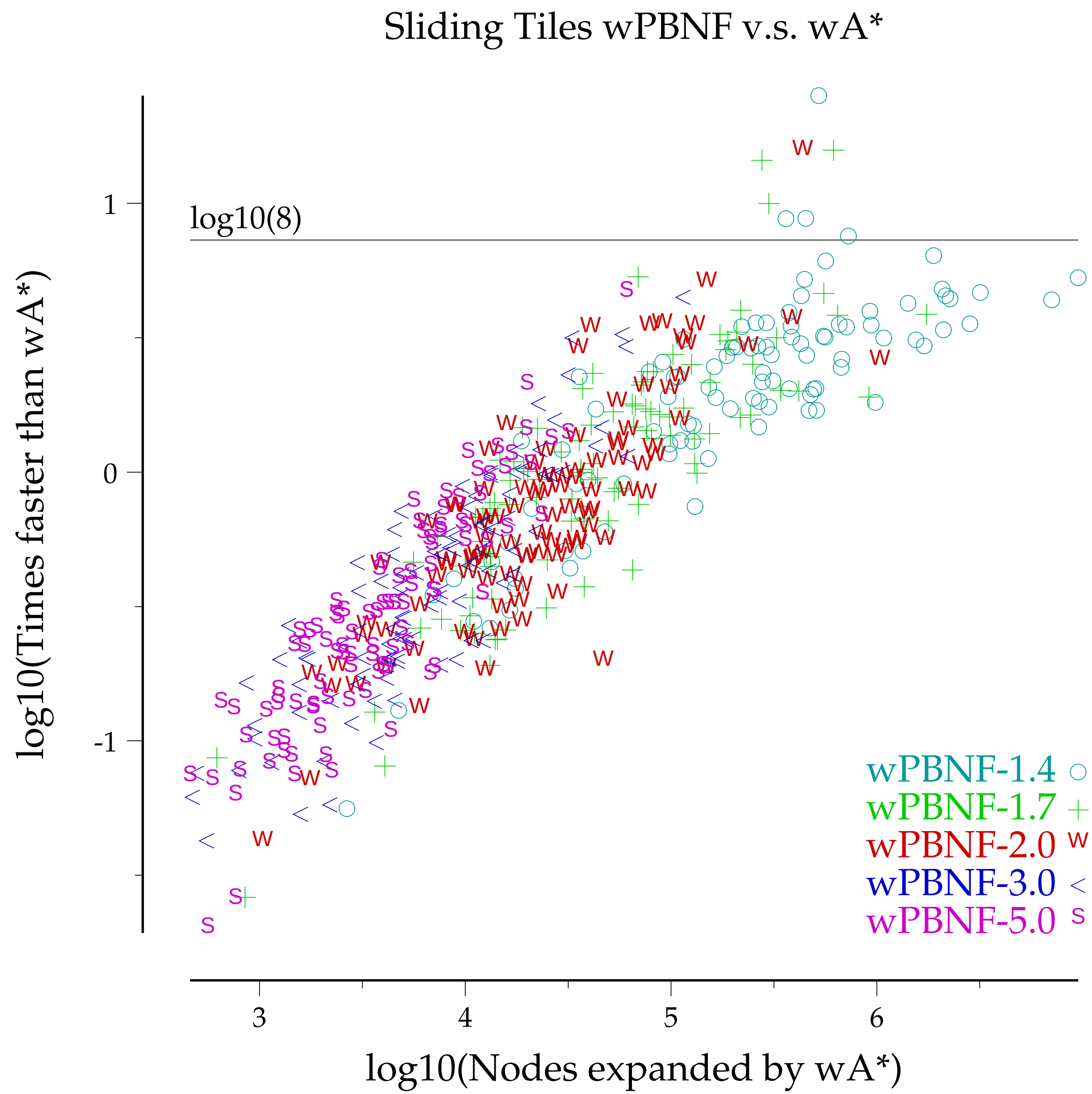
| | weight | threads 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| wPBNF | 1.4 | 0.86 | 1.40 | **2.27** | 2.01 | **2.41** | 2.48 | **2.68** | **2.58** |
| | 1.7 | 0.98 | 1.34 | 1.70 | 1.87 | 2.33 | **2.63** | 2.33 | 2.08 |
| | 2.0 | 0.96 | 1.17 | 1.45 | 1.44 | 1.57 | 1.48 | 1.56 | 1.48 |
| | 3.0 | **1.09** | 1.34 | 1.46 | 1.44 | 1.41 | 1.34 | 1.38 | 1.21 |
| | 5.0 | 0.93 | 1.04 | 1.12 | 1.04 | 1.07 | 1.13 | 0.99 | 0.92 |
| wAHDA* | 1.4 | 0.84 | 1.50 | 1.90 | **2.33** | 2.37 | 2.39 | 2.39 | 2.47 |
| | 1.7 | 0.82 | 1.42 | 1.66 | 1.90 | 1.68 | 1.75 | 1.64 | 1.70 |
| | 2.0 | 0.80 | **1.52** | 1.48 | 1.74 | 1.44 | 1.23 | 1.25 | 1.23 |
| | 3.0 | 0.75 | 1.39 | 1.30 | 1.31 | 1.10 | 0.88 | 0.73 | 0.70 |
| | 5.0 | 0.71 | 1.11 | 0.91 | 0.85 | 0.70 | 0.54 | 0.45 | 0.43 |

Speedup over serial wA

- wPBNF often gave the best performance.
- Lower weight gives more speedup.

# Benefit of Parallelism vs Difficulty





# STRIPS Planning



| | | wAPRA* | | | | wAHDA* | | | | wPBNF | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1.5 | 2 | 3 | 5 | 1.5 | 2 | 3 | 5 | 1.5 | 2 | 3 | 5 |
| 2 threads | logistics-8 | 0.99 | 1.02 | 0.59 | 1.37 | 1.25 | 1.11 | 0.80 | 1.51 | 2.68 | 2.27 | **4.06** | 1.00 |
| | blocks-16 | 1.29 | 0.88 | 4.12 | 0.30 | 1.52 | 1.09 | **4.86** | 0.38 | 0.93 | 0.54 | 0.48 | 1.32 |
| | gripper-7 | 0.76 | 0.76 | 0.77 | 0.77 | 1.36 | 1.35 | 1.33 | 1.30 | 2.01 | 1.99 | 1.99 | **2.02** |
| | satellite-6 | 0.68 | 0.93 | 0.70 | 0.75 | 1.15 | 1.09 | 1.28 | 1.44 | 2.02 | 1.53 | **5.90** | 3.04 |
| | elevator-12 | 0.65 | 0.72 | 0.71 | 0.77 | 1.16 | 1.20 | 1.27 | 1.22 | 2.02 | 2.08 | **2.21** | 2.15 |
| | freecell-3 | 1.03 | 1.00 | 1.78 | 1.61 | 1.49 | 1.20 | 7.56 | 1.40 | 2.06 | 0.84 | 8.11 | **10.69** |
| | depots-13 | 0.73 | 1.25 | 0.97 | 1.08 | 0.92 | 1.29 | 0.96 | 1.09 | 2.70 | **4.49** | 0.82 | 0.81 |
| | driverlog-11 | 0.91 | 0.79 | 0.94 | 0.93 | **1.30** | 0.97 | 0.96 | 0.93 | 0.85 | 0.19 | 0.69 | 0.62 |
| | gripper-8 | 0.63 | 0.61 | 0.62 | 0.62 | 1.14 | 1.16 | 1.15 | 1.16 | 2.06 | 2.04 | **2.08** | 2.07 |
| 7 threads | logistics-8 | 3.19 | 3.10 | 3.26 | 2.58 | 4.59 | 4.60 | 3.61 | 2.58 | **7.10** | 6.88 | 1.91 | 0.46 |
| | blocks-16 | 3.04 | 1.37 | 1.08 | 0.37 | **3.60** | 1.62 | 0.56 | 0.32 | 2.87 | 0.70 | 0.37 | 1.26 |
| | gripper-7 | 1.71 | 1.74 | 1.73 | 1.82 | 3.71 | 3.66 | 3.74 | 3.83 | **5.67** | 5.09 | 5.07 | 5.18 |
| | satellite-6 | 1.11 | 1.01 | 1.29 | 1.44 | 3.22 | 3.57 | 3.05 | 3.60 | 4.42 | 2.85 | 2.68 | **5.89** |
| | elevator-12 | 0.94 | 0.97 | 1.04 | 1.02 | 2.77 | 2.88 | 2.98 | 3.03 | 6.32 | 6.31 | 6.60 | **7.10** |
| | freecell-3 | 3.09 | 7.99 | 2.67 | 2.93 | 4.77 | 2.71 | 48.66 | 4.77 | 7.01 | 2.31 | 131.12 | **1,721.33** |
| | depots-13 | 2.38 | 5.36 | 1.13 | 1.17 | 2.98 | **6.09** | 1.22 | 1.17 | 3.12 | 1.80 | 0.87 | 0.88 |
| | driverlog-11 | 1.90 | 1.25 | 0.93 | 0.92 | **3.52** | 1.48 | 0.95 | 0.92 | 1.72 | 0.43 | 0.67 | 0.42 |
| | gripper-8 | 1.70 | 1.68 | 1.68 | 1.74 | 3.71 | 3.63 | 3.67 | 4.00 | **5.85** | 5.31 | 5.40 | 5.44 |

Speedup over serial wA*

- Most **red** is under wPBNF (13 of 18).
- Blue is everywhere.

# Summary of Bounded Suboptimal Results



- In general speedup was not as good as optimal search.
  - Some harder problems gave excellent speedup.
- Lower weights can increase benefit of parallelizing.

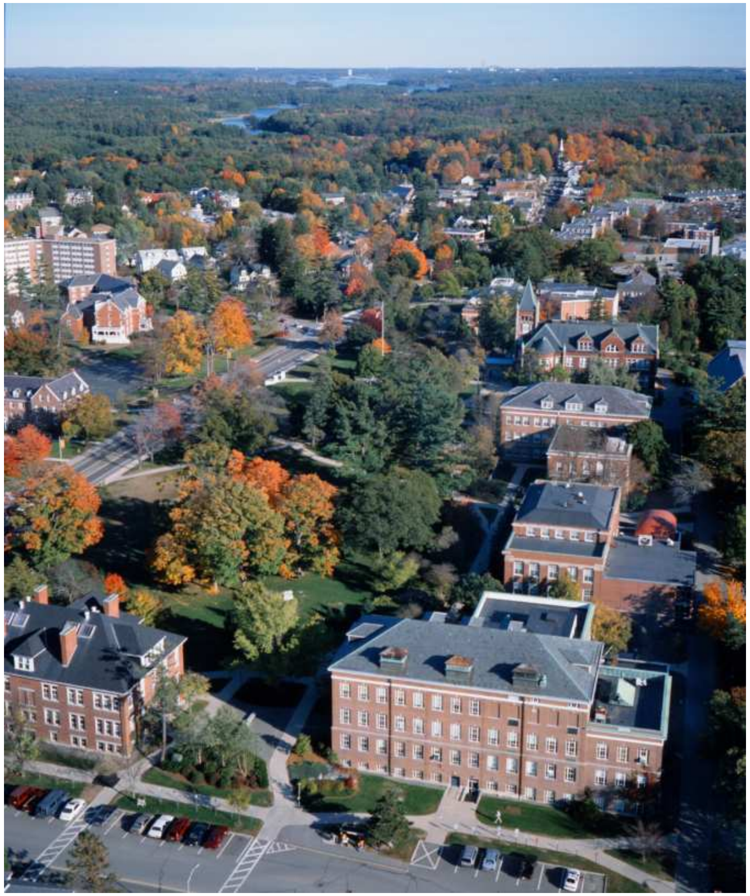
# Conclusion

# Conclusion



- Parallel search can make your programs run faster today.
  - Multicore is not going away.
  - Email me for the code (C++): burns.ethan@gmail.com
- PBNF and PRA* are simple and general.
  - Easily extendable to suboptimal (and anytime) search.
  - PBNF generally performed better than the other algorithms tested.
- Abstraction is beneficial for parallel search.
- Parallel search is more beneficial on harder problems.

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, systems