Parallel Best-First Search: Optimal and Suboptimal Solutions

Ethan Burns, Seth Lemons, Wheeler Ruml

Rong Zhou

University of New Hampshire

Palo Alto Research Center

Supported in part by NSF grant IIS-0812141.

Motivation: The Future is Multicore

Now we're into the explicit parallelism multiprocessor era, and this will dominate for the foreseeable future. I don't see any technology or architectural innovation on the horizon that might be competitive with this approach.

John Hennessy
President of Stanford University,
Cofounder of MIPS Computer Systems

(*A Conversation with John Hennessy and David Patterson*, ACM Queue, December 2006)

Overview

- Previous: Parallel Structured Duplicate Detection (Zhou and Hansen, 2007)
  - Parallelized breadth-first search.
  - Used abstraction to divide labor between threads.

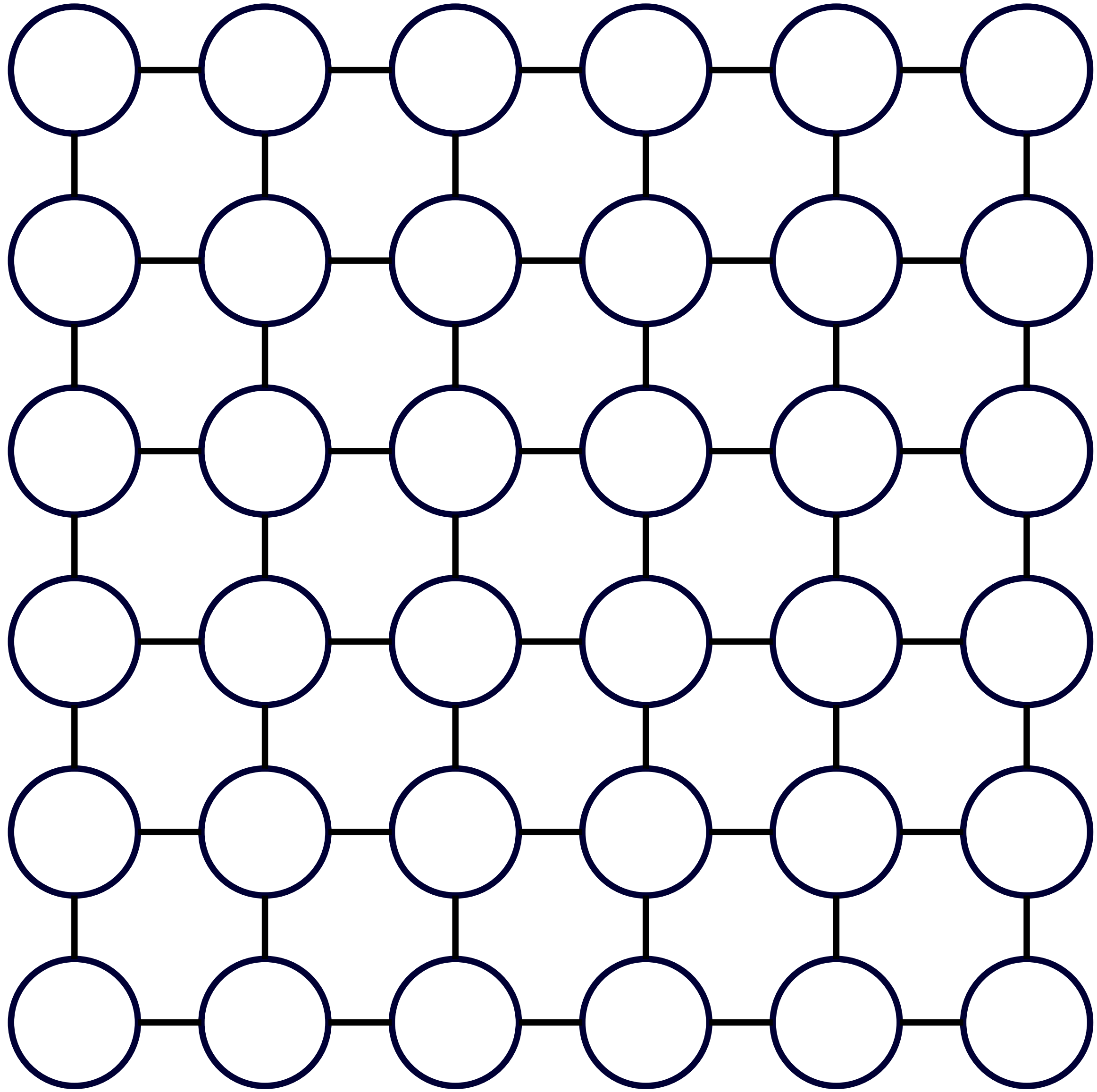
Overview

- Previous: Parallel Structured Duplicate Detection (Zhou and Hansen, 2007)
  - Parallelized breadth-first search.
  - Used abstraction to divide labor between threads.
- New: Parallel Best NBlock First Search
  - Approximates best-first ordering.
  - Requires care to avoid livelock.
  - Broadly applicable framework.

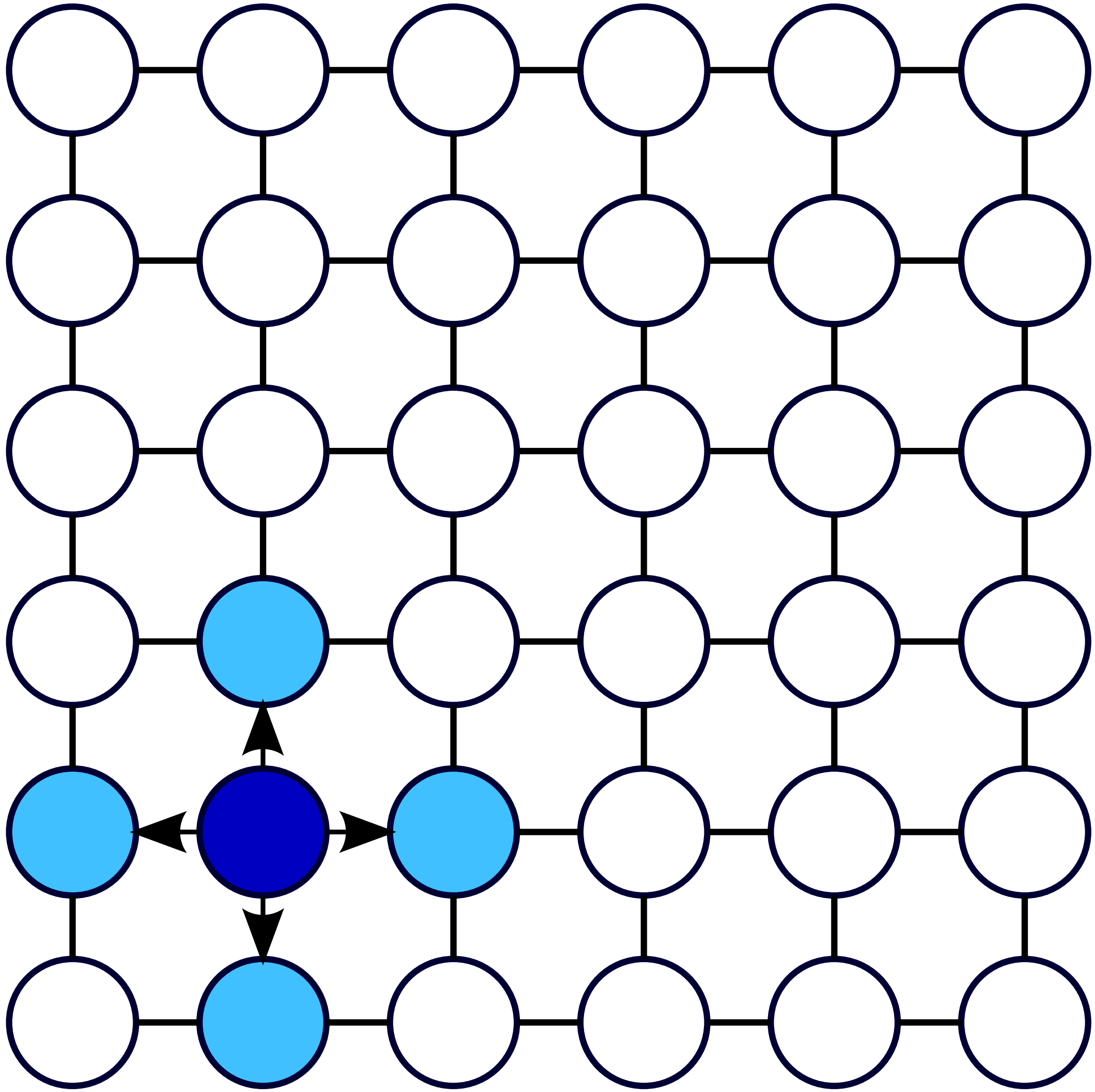
Overview

- Previous: Parallel Structured Duplicate Detection (Zhou and Hansen, 2007)
  - Parallelized breadth-first search.
  - Used abstraction to divide labor between threads.
- New: Parallel Best NBlock First Search
  - Approximates best-first ordering.
  - Requires care to avoid livelock.
  - Broadly applicable framework.
- New: Two Pruning Rules for Parallel Suboptimal Search
  - Pruning against an incumbent.
  - Pruning some duplicates.

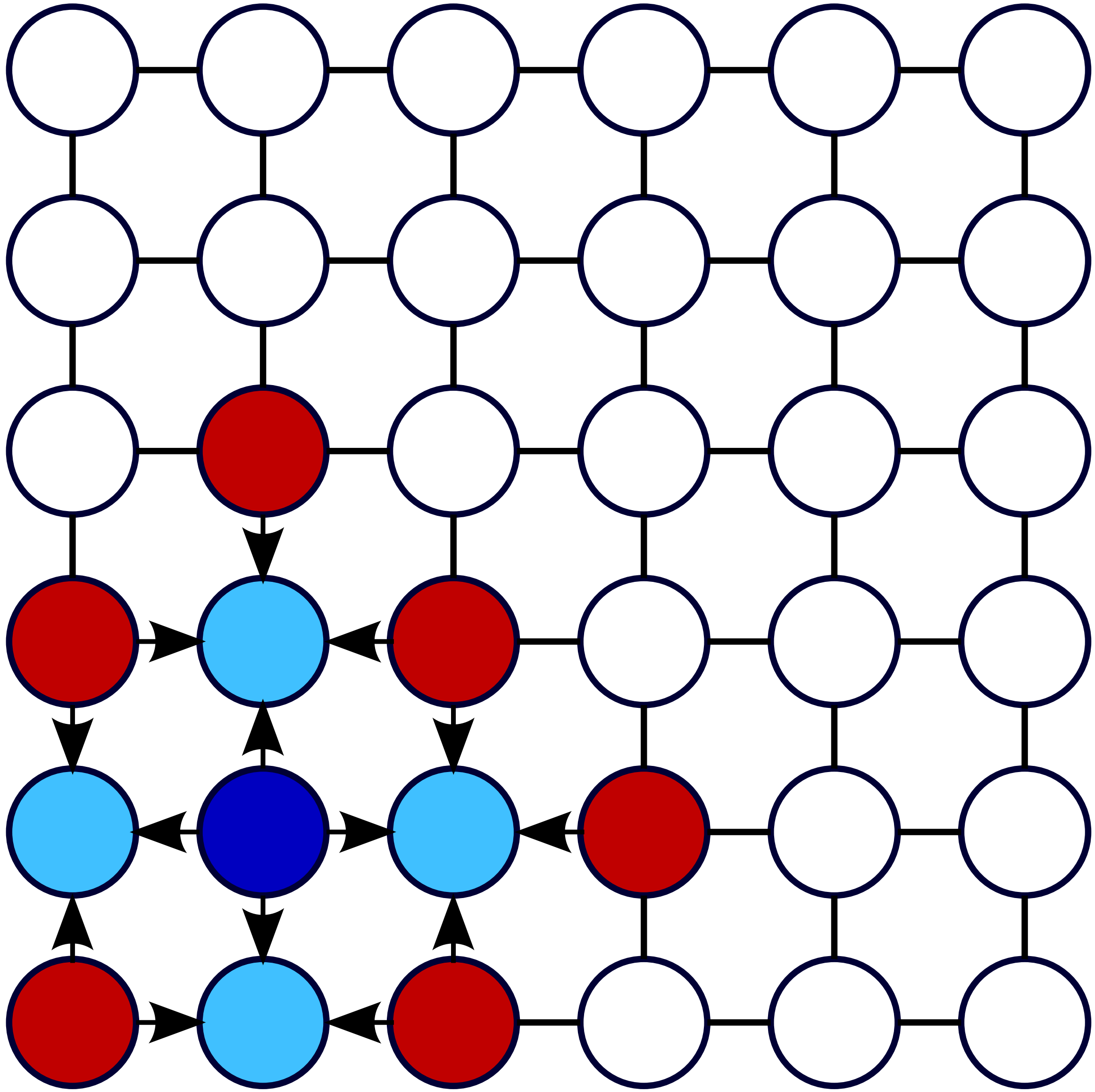
Naive Parallel Search



Naive Parallel Search

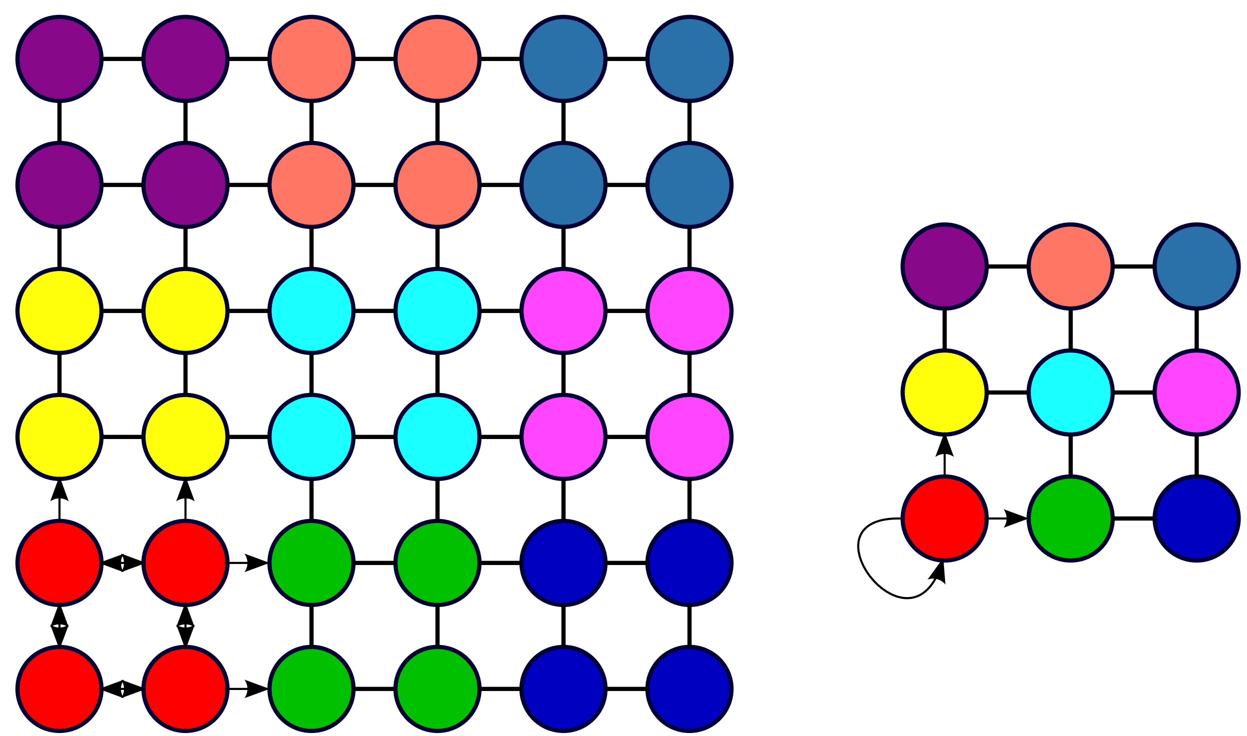


Naive Parallel Search



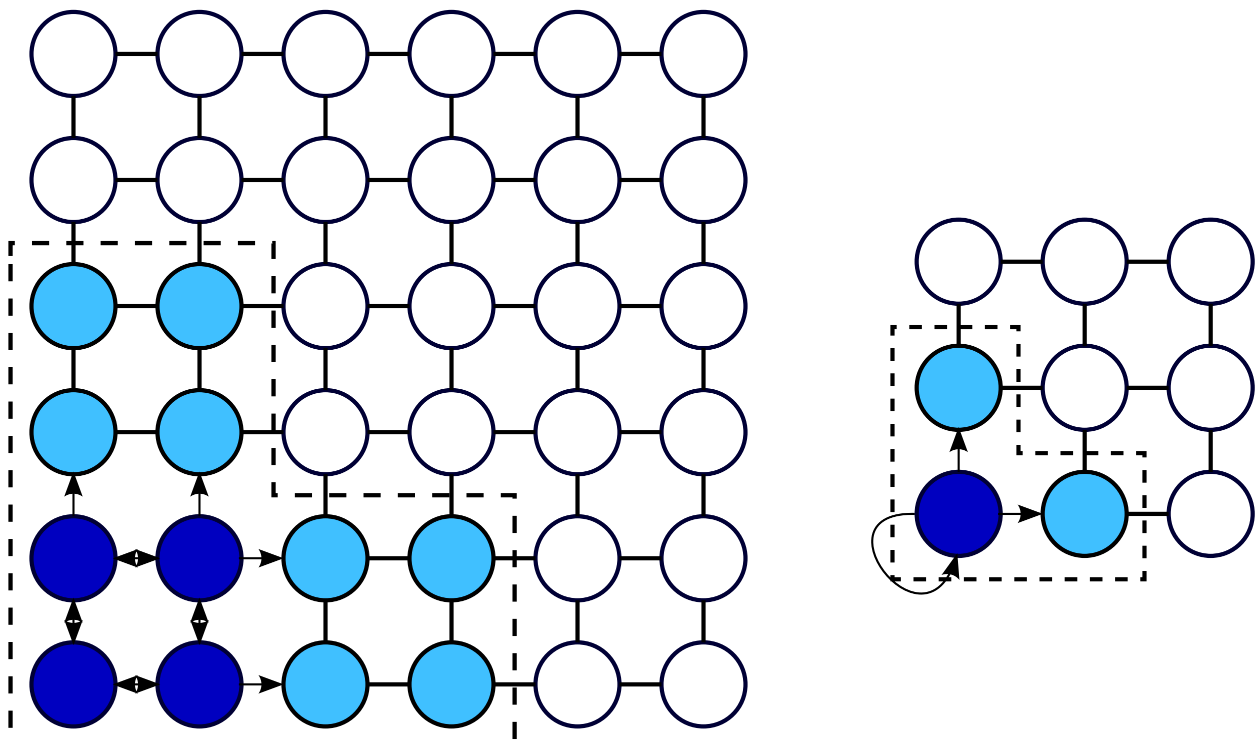
Division of Labor by Abstraction

- Work is divided among threads using a special hash function based on abstraction.
  - Few possible locations for children.



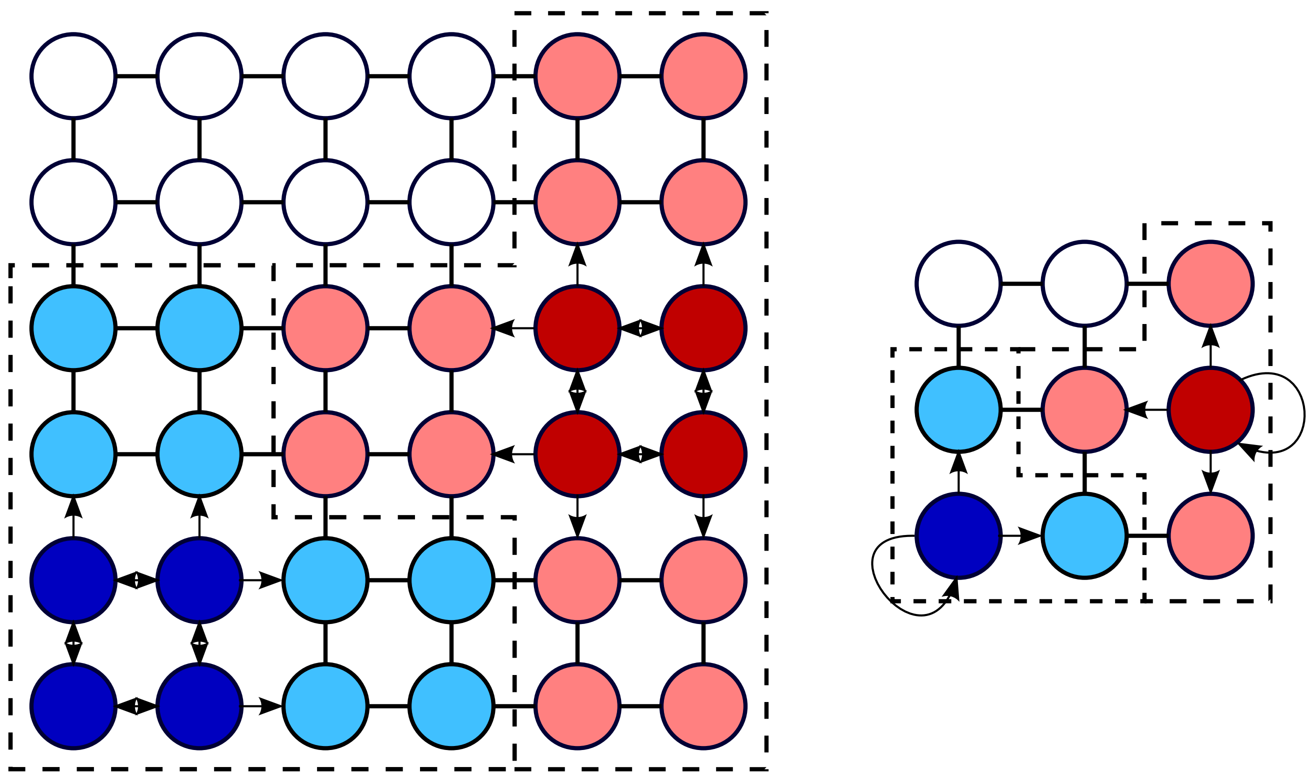
Division of Labor by Abstraction

- Work is divided among threads using a special hash function based on abstraction.
  - Threads acquire groups of states called $n$blocks.
  - Exclusive access to a duplicate detection scope.



Division of Labor by Abstraction

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



New: Parallel Best NBlock First Search

Parallel Best NBlock First Search

1. Search disjoint $n$blocks in parallel.
   - Maintain a *heap of free $n$blocks*.
   - Greedily acquire best free $n$block (and its scope).
   - Each $n$block is searched in $f(n)$ order.

Parallel Best NBlock First Search

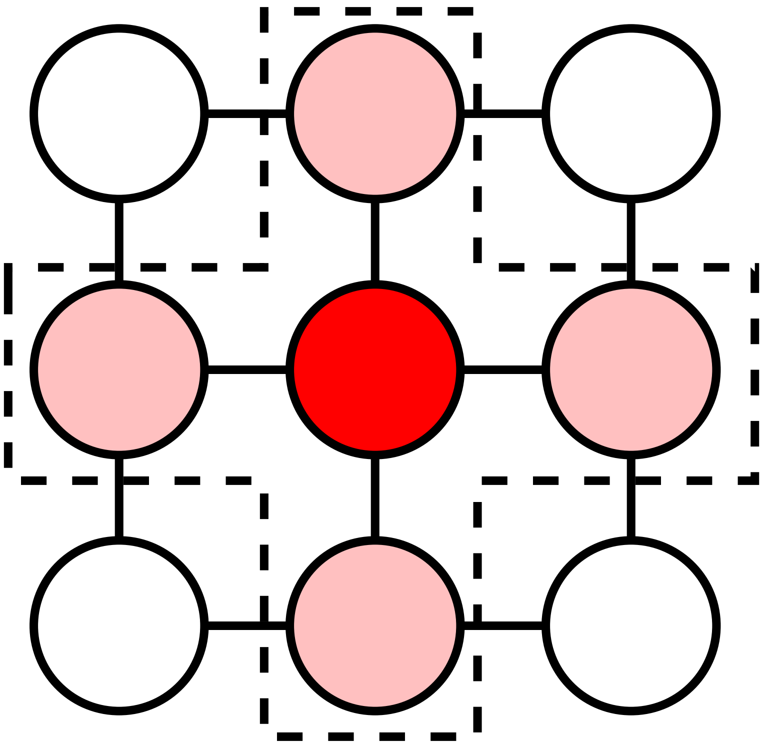
1. Search disjoint $n$blocks in parallel.
   - Maintain a *heap of free $n$blocks*.
   - Greedily acquire best free $n$block (and its scope).
   - Each $n$block is searched in $f(n)$ order.
2. Switch $n$blocks when a better one becomes free.
   - Approximates best-first expansion order.
   - Perform a minimum amount of work before switching.

Parallel Best NBlock First Search

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

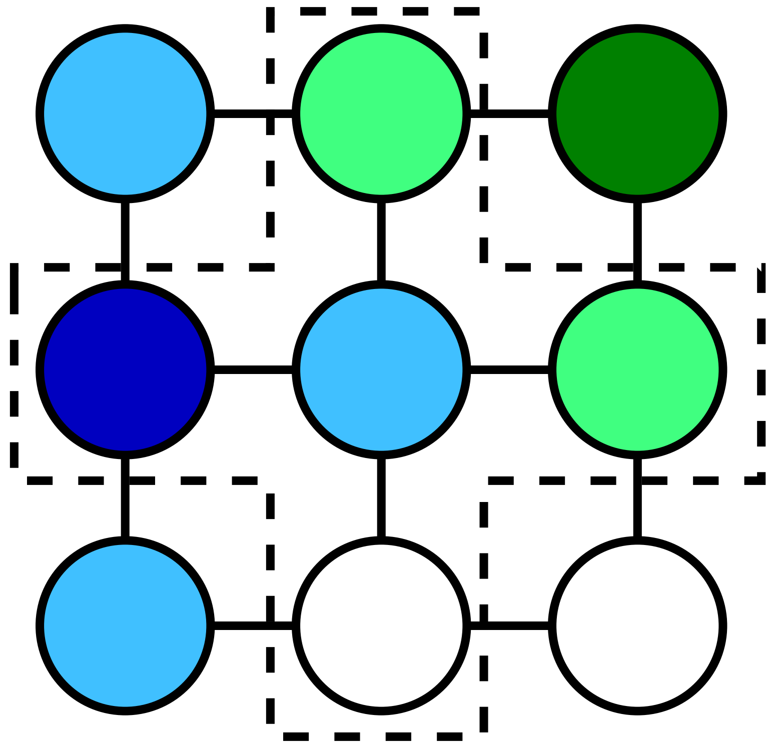
“Safe” PBNF

- Problem with best free $n$block ordering:



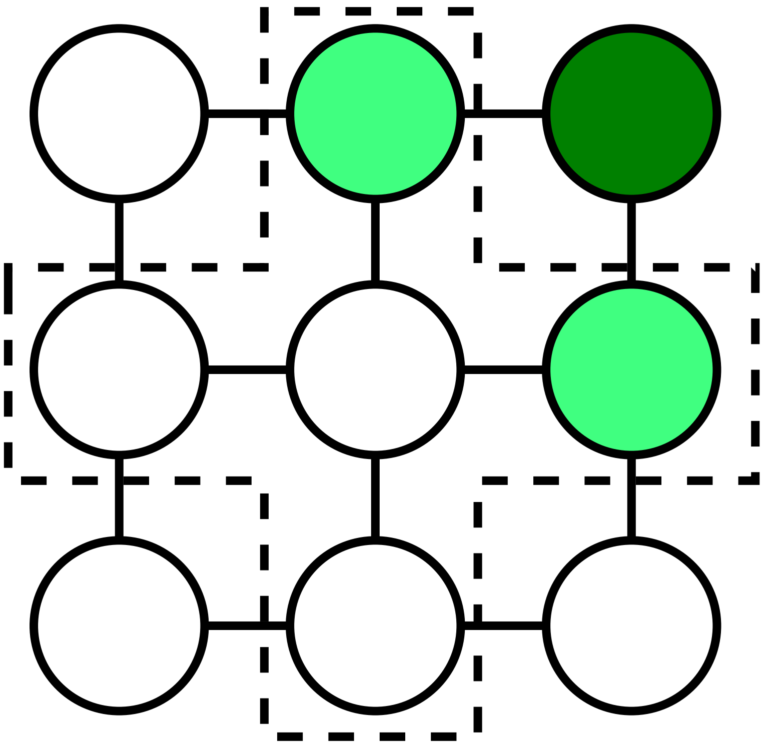
“Safe” PBNF

- Problem with best free $n$block ordering:

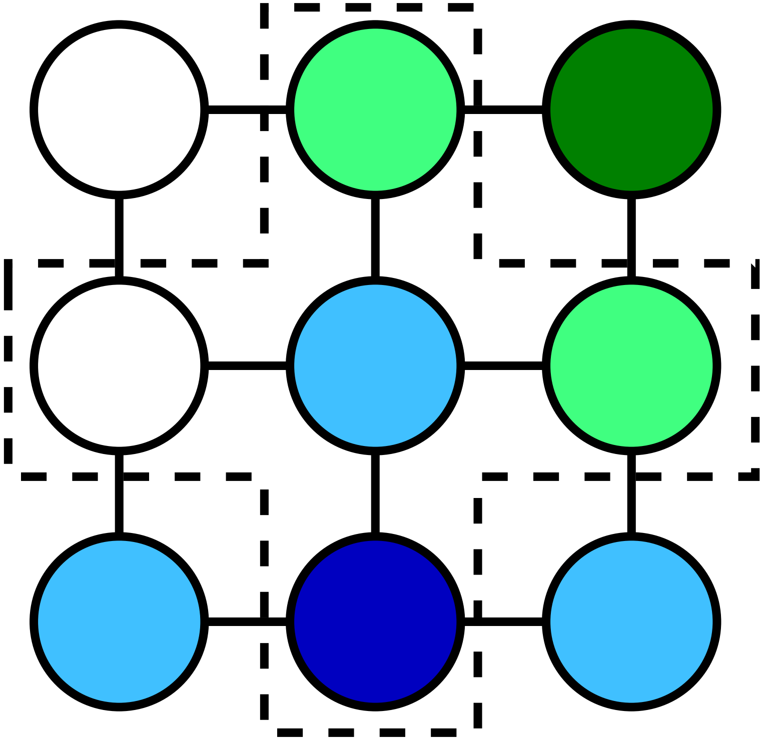


“Safe” PBNF

- Problem with best free $n$block ordering:



- Problem with best free $n$block ordering:

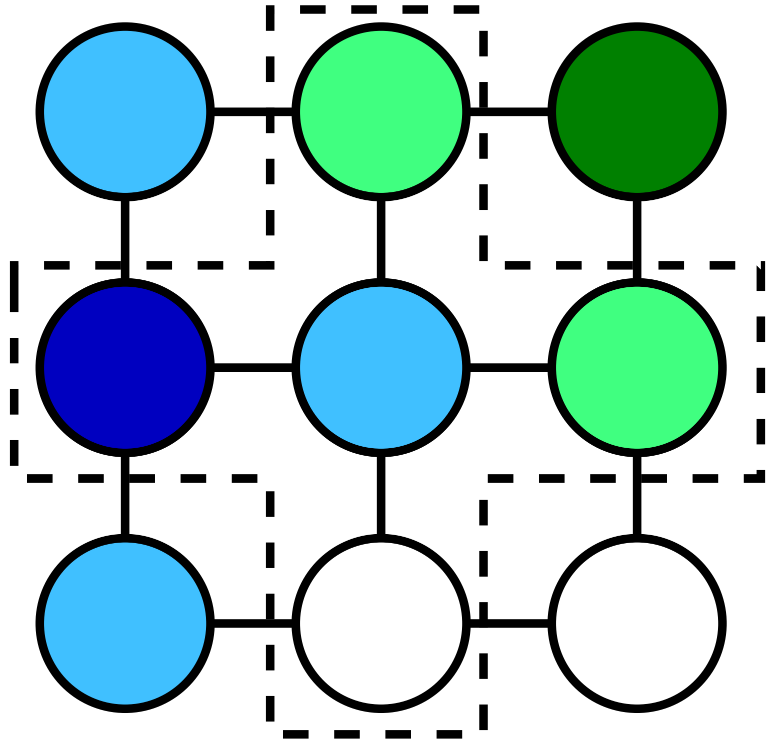


“Safe” PBNF

- **No guarantee that a given $n$block will become free.**
  - In infinite search spaces, there can be livelock.
- Solution: check for *hot* $n$blocks
  - Flag better $n$blocks as *hot*
  - Release an $n$block to free an interfered hot $n$block.
- Solution:

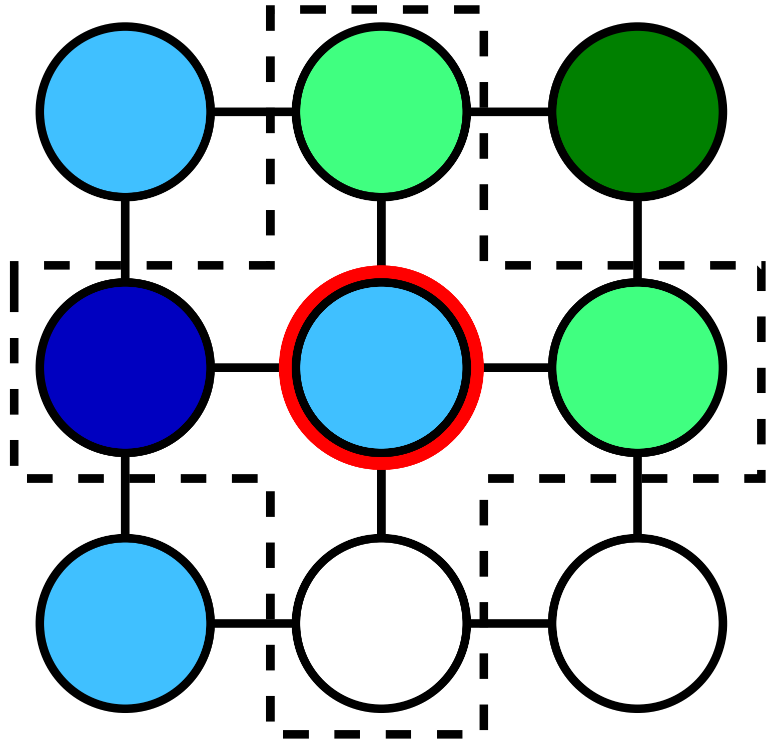
“Safe” PBNF

- Solution:

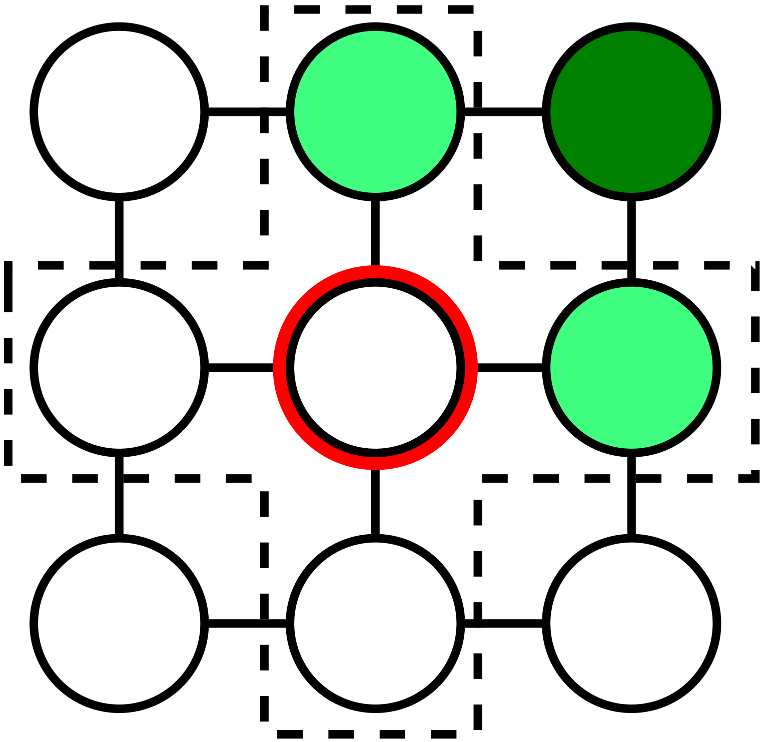


“Safe” PBNF

- Solution:

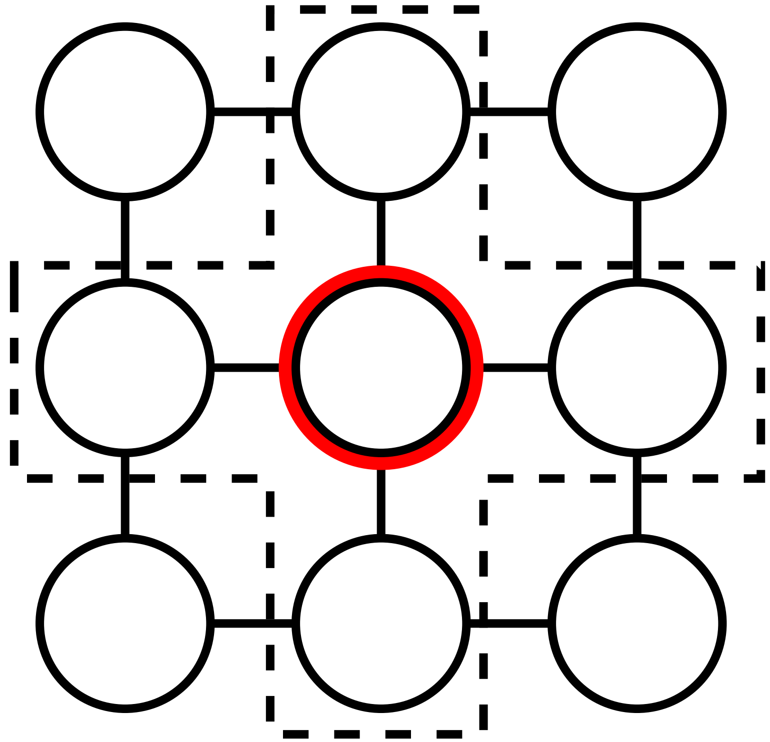


- Solution:



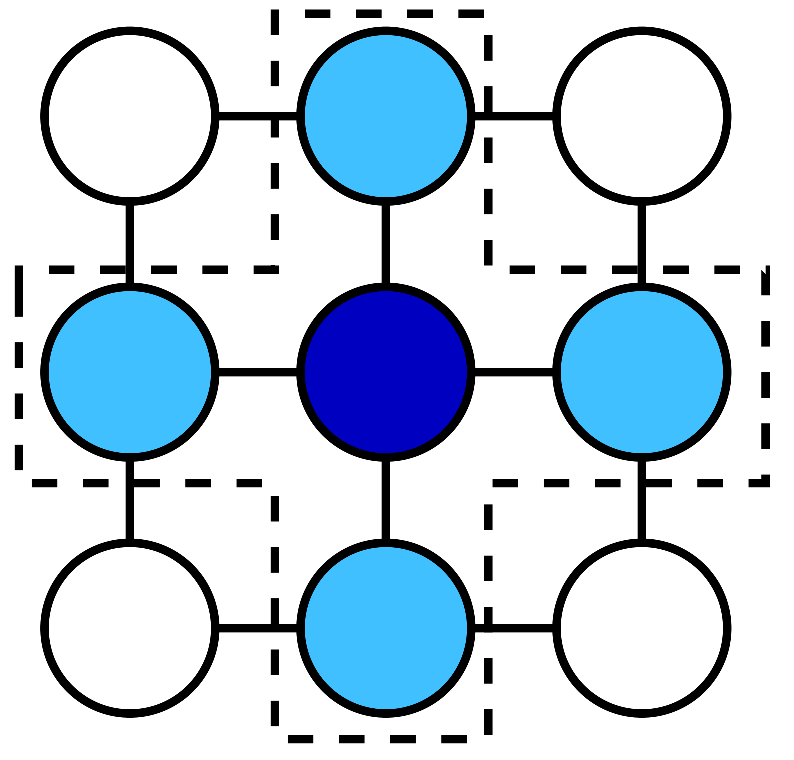
“Safe” PBNF

- Solution:



“Safe” PBNF

- Solution:



Empirical Evaluation

Software

- C++, POSIX threads, jemalloc library
- Fedora 9

Hardware

- Dual quad-core Intel Xeon E5320 1.86GHz 64-bits
- 16Gb RAM

Domains

- Grid pathfinding
  - Abstraction: courser grid
- 15 puzzles
  - Abstraction: ignore some tile numbers
- STRIPS planning
  - Abstraction: generated automatically

Algorithms

PA*

- Basic A* with a lock on open and closed lists.

Lock-free PA*

- PA* with lock-free data structures.

KBFS (Felner et al., 2003)

- Expand the K best open nodes in parallel.

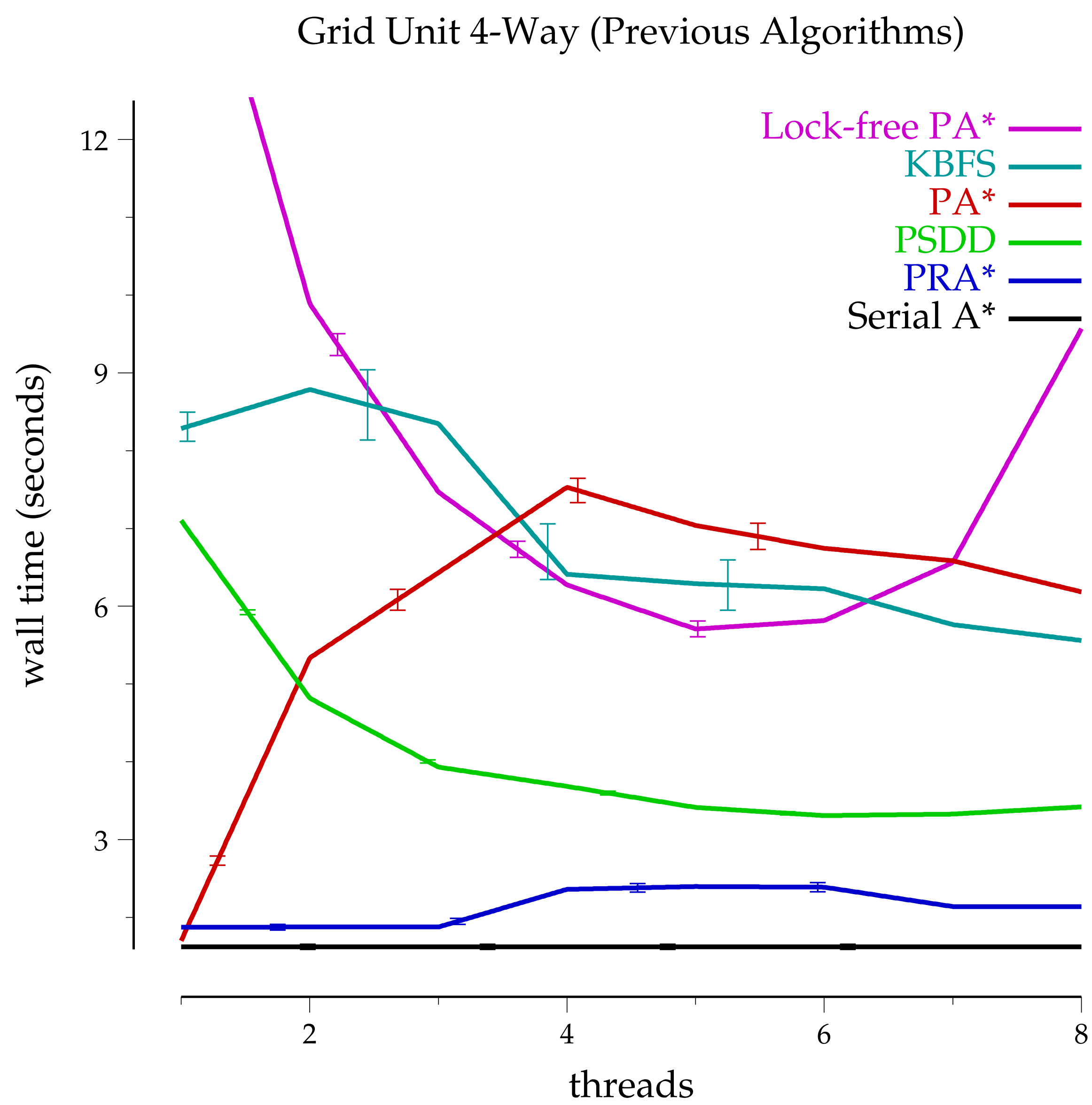
PRA* (Evett et al., 1995)

- Hash nodes to distribute among processors.
- Synchronized message queues for “incoming” nodes.

IDPSDD

- PSDD with iterative-deepening for bounds.

Four-way Grid Pathfinding (Previous Algorithms)



APRA*

- PRA* with a novel abstraction based hashing.

PSDD (Zhou and Hansen, 2007)

- Abstraction to find disjoint portions of a search space.
- Breadth-first search
- All threads synchronize at each layer

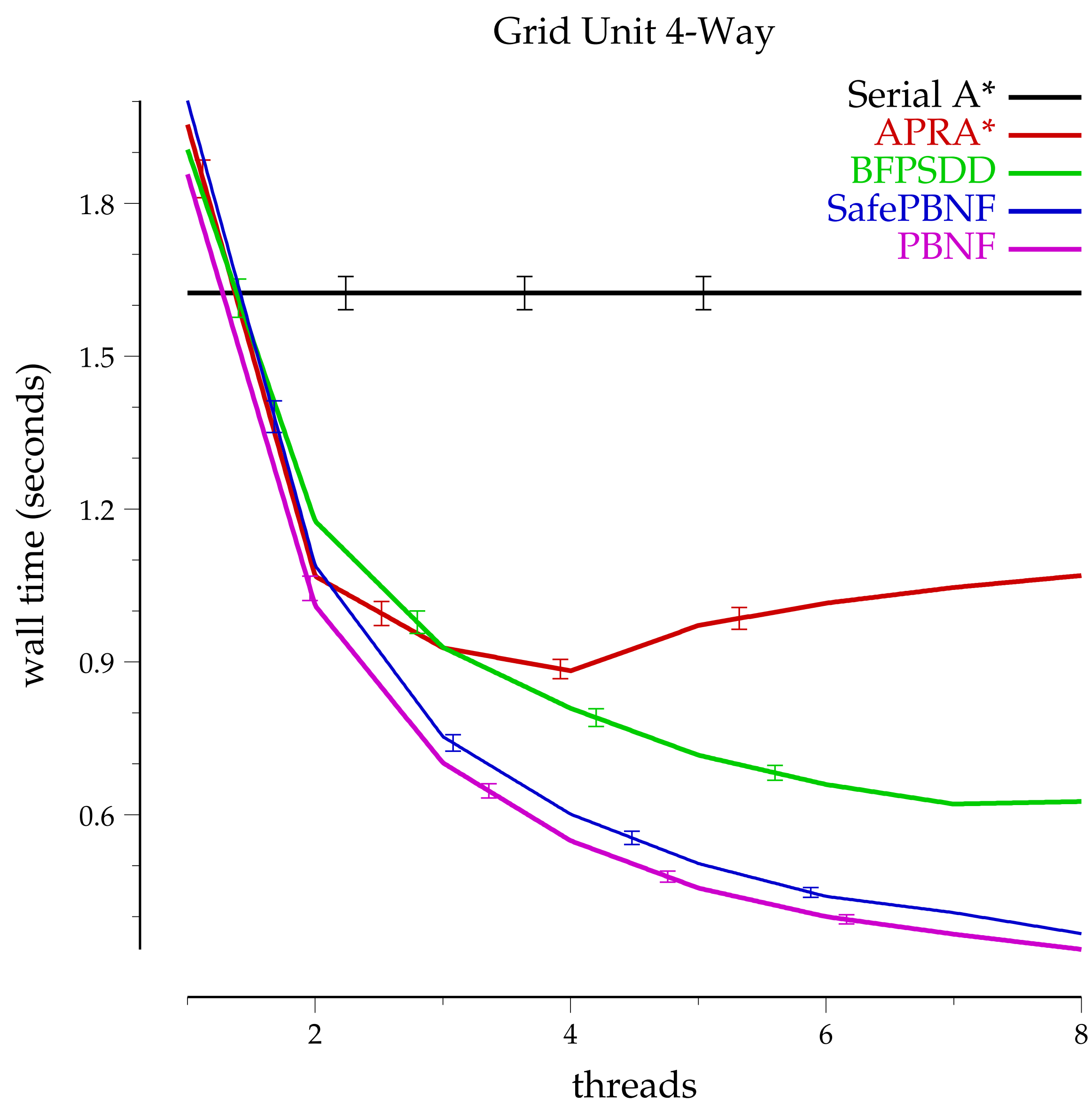
BFPSDD

- PSDD with $f(n)$ layers instead of depth layers.

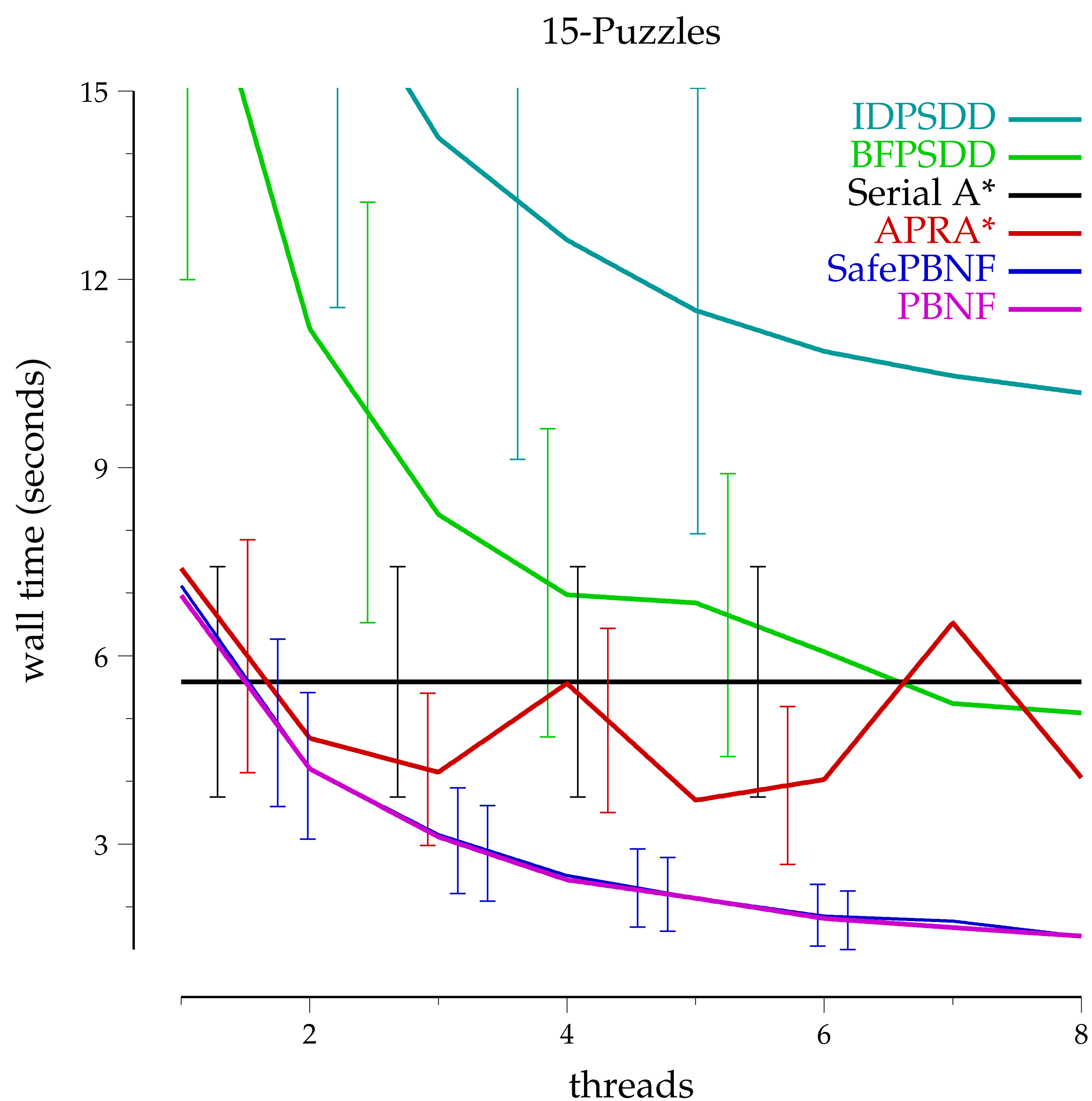
(Safe) PBNF

- Acquire the best free $n$block (prevent live-locks).

Four-way Grid Pathfinding (Optimal)



Easy Sliding 15-Puzzles (Optimal)



STRIPS Planning

	threads	logistics-6	blocks-14	gripper-7	satellite-6	elevator-12	freecell-3	depots-7	driverlog-11	gripper-8
A*	1	2.3	5.2	118	131	336	199	M	M	M
APRA*	1	1.5	7.1	60	96	213	150	301	322	528
	3	0.76	5.5	51	49	269	112	144	103	M
	5	1.2	3.8	41	66	241	61	M	M	M
	7	0.84	3.7	28	49	169	40	M	M	M
PNBF	1	1.3	6.3	40	68	157	186	M	M	230
	3	0.72	3.8	16	34	56	64	M	M	96
	5	0.58	2.7	11	21	35	44	M	M	61
	7	**0.53**	2.6	**8.6**	17	**27**	**36**	M	M	**48**
SafePBNF	1	1.2	6.2	40	77	150	127	156	154	235
	3	0.64	2.7	17	24	54	47	63	60	98
	5	0.56	2.2	11	17	34	38	43	39	64
	7	0.62	**2.0**	9.2	**14**	**27**	37	**35**	**31**	52
BFPSDD	1	2.1	7.8	42	62	152	131	167	152	243
	3	1.1	4.3	18	24	59	57	67	62	101
	5	0.79	3.9	12	20	41	48	48	43	71
	7	0.71	3.4	10	**14**	32	45	43	35	59

Time in seconds

Summary

- PBNF gives the best performance and scalability across all domains tested.
- Overhead of livelock prevention procedure seems low.
- Simple to apply to:
  - non-unit cost domains.
  - any $f(n)$ ordering.

Bounded Suboptimal Search

Weighted PBNF

- Sort open lists on $f'(n) = g(n) + w \cdot h(n)$.
- Terminate when open list is empty or minimum $f'(n) \geq g(s)$ for incumbent solution s.
- Solution returned guaranteed to be within w factor of optimal.

Pruning When Search Order is not Strictly Best-First

- wA* can drop duplicates while maintaining bounded suboptimality. We can't.

Pruning When Search Order is not Strictly Best-First

- wA* can drop duplicates while maintaining bounded suboptimality. We can't.
- Rule 1: Only re-expand duplicates if the new path is a factor w better.

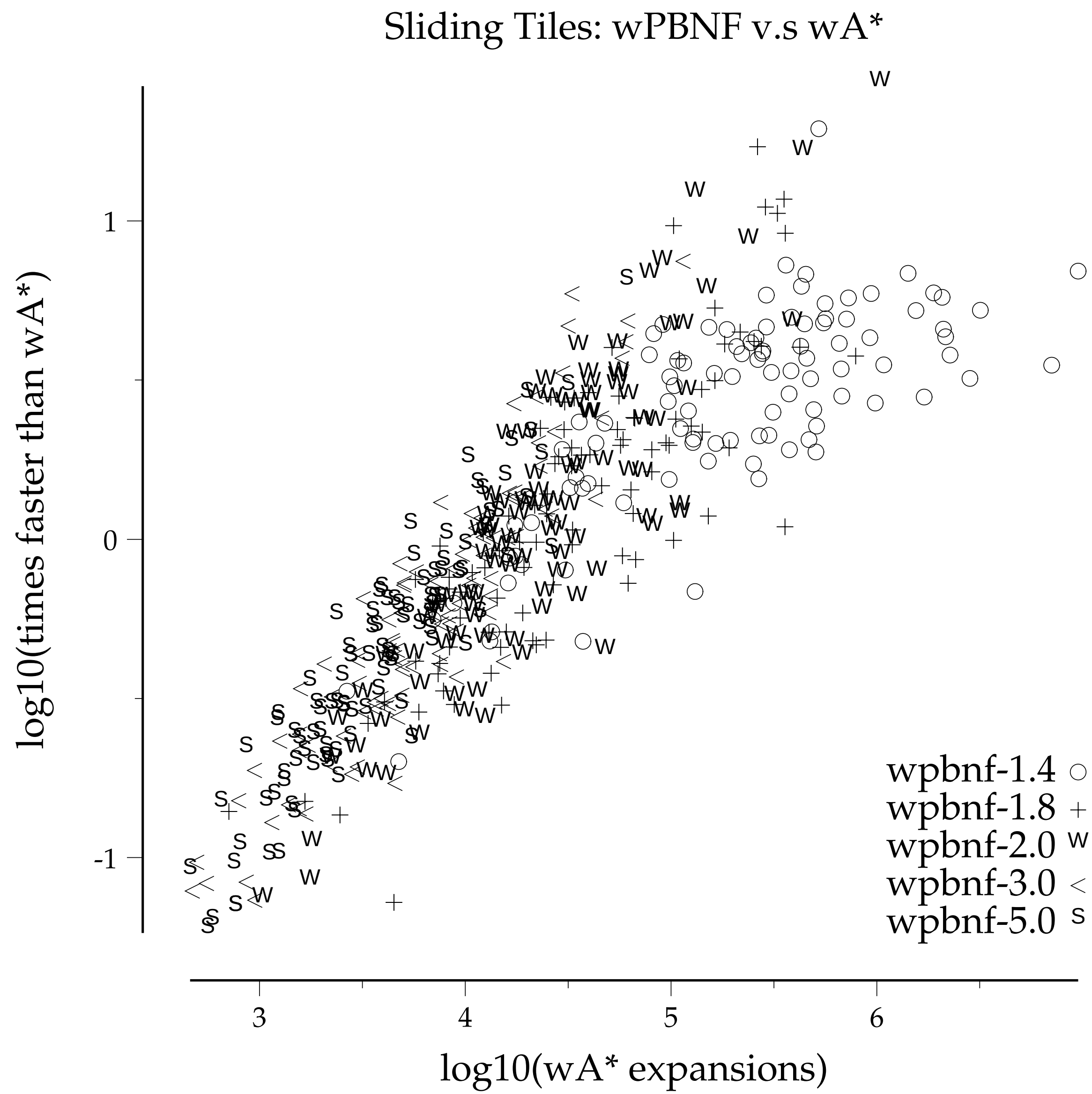
Pruning When Search Order is not Strictly Best-First

- wA* can drop duplicates while maintaining bounded suboptimality. We can't.
- Rule 1: Only re-expand duplicates if the new path is a factor w better.
- Rule 2: Only retain nodes a factor of w better than incumbent. (See Burns et al in ICAPS '09 for details.)

Easy Sliding 15-Puzzles (Bounded Suboptimal)

	wPBNF					wBFPSDD					wAPRA*				
w	1	2	4	5	8	1	2	4	5	8	1	2	4	5	8
1.4	0.65	1.12	1.65	1.92	**2.62**	0.64	0.87	1.07	1.32	1.54	0.63	1.03	0.90	1.70	1.35
1.7	0.41	0.76	1.37	1.50	1.49	0.60	0.76	0.99	1.06	1.14	0.61	0.97	0.98	**1.79**	1.06
2.0	0.37	0.62	1.10	1.34	**1.46**	0.43	0.47	0.62	0.63	0.66	0.61	1.23	0.82	1.37	0.96
3.0	0.74	0.62	**0.90**	0.84	0.78	0.34	0.39	0.46	0.42	0.32	0.57	0.86	0.54	0.68	0.45

All Korf Sliding 15-Puzzles (Bounded Suboptimal)



STRIPS Planning (Bounded Suboptimal)

		wPBNF				wBFPSDD				wAPRA*			
		1.5	2	3	5	1.5	2	3	5	1.5	2	3	5
2 Threads	logistics-8	2.68	2.27	**4.06**	1.00	1.86	2.12	1.14	0.86	1.01	0.98	0.64	1.14
	blocks-16	0.93	0.54	0.48	1.32	0.34	0.19	0.16	**2.42**	1.16	0.82	**2.42**	0.25
	gripper-7	2.01	1.99	1.99	**2.02**	1.91	1.89	1.86	1.84	0.77	0.76	0.76	0.75
	satellite-6	2.02	1.53	5.90	3.04	1.71	2.22	**7.50**	2.80	0.71	0.84	0.67	0.79
	elevator-12	2.02	2.08	**2.21**	2.15	1.76	1.76	1.81	2.18	0.70	0.68	0.72	0.73
	freecell-3	2.06	0.84	8.11	10.7	1.42	0.54	16.9	**55.8**	1.12	1.18	0.84	1.03
	depots-13	2.70	**4.49**	0.82	0.81	1.48	1.58	0.18	0.16	0.74	1.12	0.32	0.29
	driverlog-11	0.85	0.19	0.69	0.62	0.85	0.11	0.19	0.21	**0.86**	0.33	0.20	0.19
	gripper-8	2.06	2.04	**2.08**	2.07	2.00	1.96	1.97	1.98	0.61	M	M	M
7 Threads	logistics-8	**7.10**	6.88	1.91	0.46	3.17	3.59	0.11	0.1	3.17	2.77	2.23	1.02
	blocks-16	**2.87**	0.7	0.37	1.26	0.49	0.22	0.01	0.32	2.67	1.06	1.02	0.13
	gripper-7	**5.67**	5.09	5.07	5.18	4.33	4.28	4.14	4.05	1.76	1.75	1.74	1.80
	satellite-6	4.42	2.85	2.68	**5.89**	3.13	2.31	3.01	1.05	1.10	0.96	1.18	0.97
	elevator-12	**6.32**	6.31	7.10	3.68	3.78	4.04	3.95	0.94	0.96	1.03	1.01	1.01
	freecell-3	7.01	2.31	131	**1721**	2.12	0.70	44.5	137	2.48	0.76	2.93	2.96
	depots-13	**3.12**	1.80	0.87	0.88	1.88	1.87	0.13	0.12	M	2.58	0.12	0.11
	driverlog-11	**1.72**	0.43	0.67	0.42	1.26	0.21	0.30	0.23	M	M	M	M
	gripper-8	**5.85**	5.31	5.40	5.44	4.62	4.55	4.55	4.51	M	M	M	M

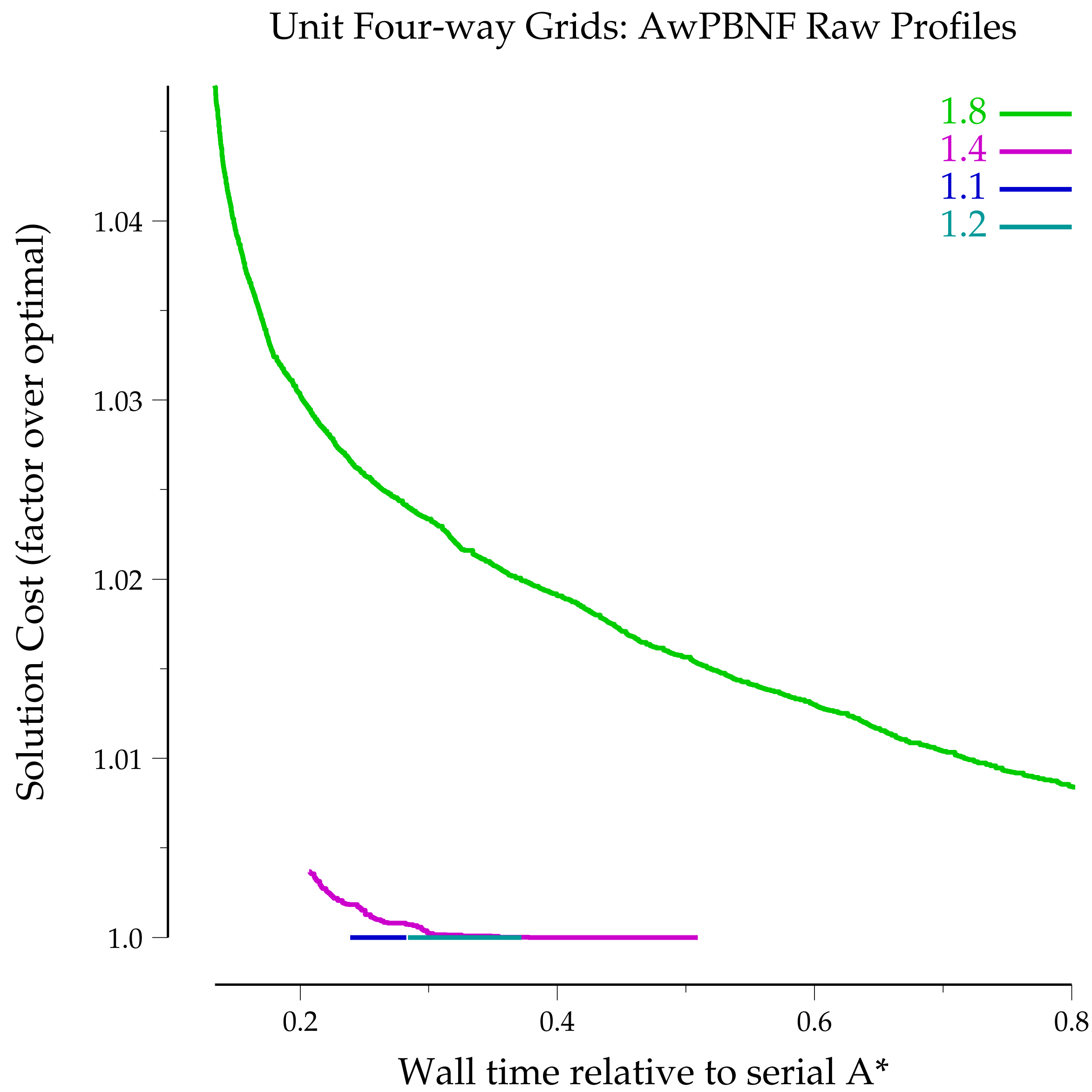
Summary

- Higher weights reduce benefit of parallelizing.
- Greater benefit on problems that require more work for wA*.

Anytime Search

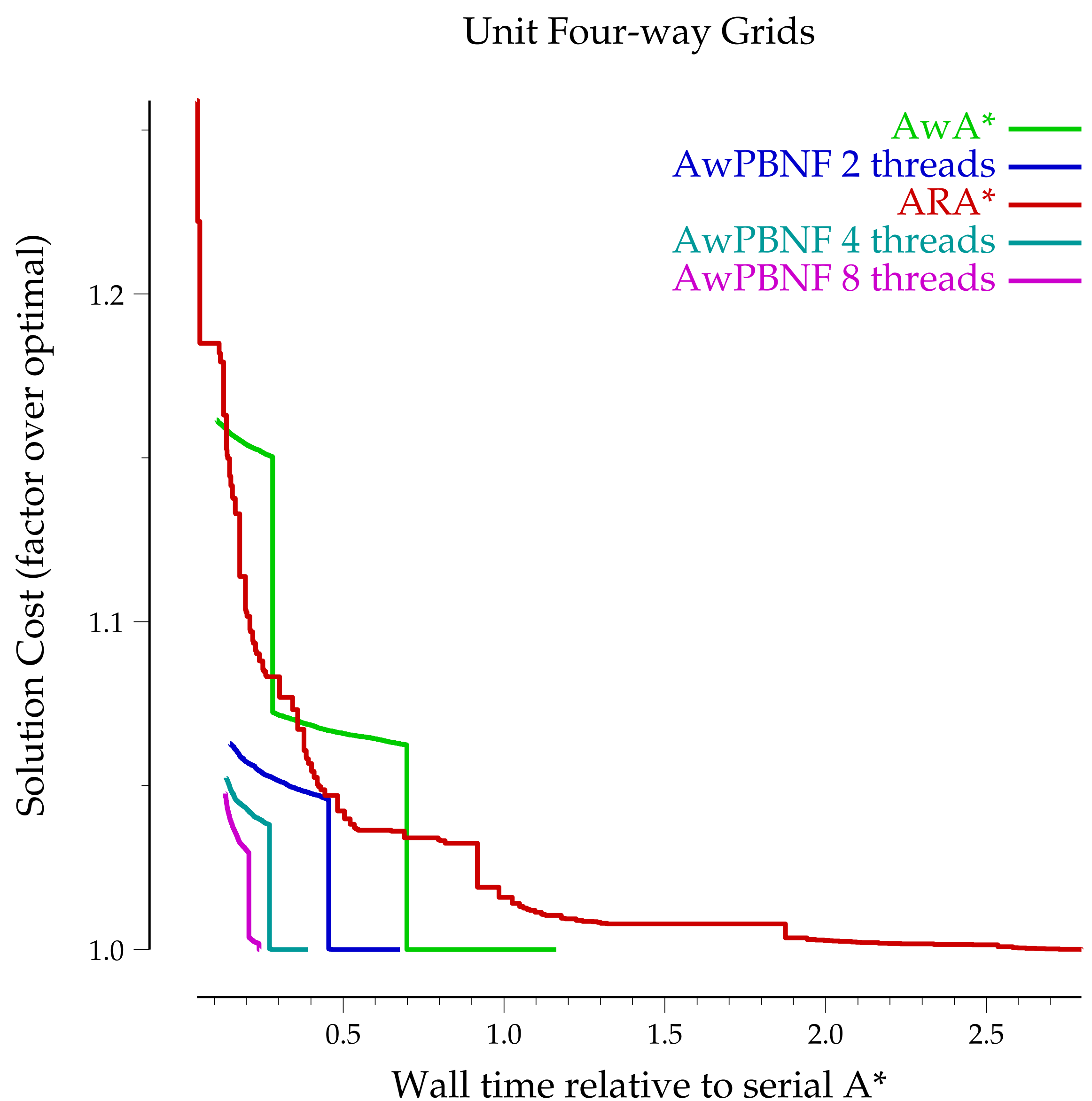
Anytime Weighted PBNF

- Sort open lists on $f'(n) = g(n) + w \cdot h(n)$.
- Continue after first goal, returning stream of improving solutions. (Hansen and Zhou, 2007)
- Terminate when open list is empty or minimum $f(n) \geq g(s)$ for incumbent solution s.
- Converges to optimal solution, proving optimality.



Unit Four-way Grids: AwPBNF Raw Profiles
1.8
1.4
1.1
1.2
Solution Cost (factor over optimal)
1.04
1.03
1.02
1.01
1.0
0.2
0.4
0.6
0.8
Wall time relative to serial A*

Four-way Grid Pathfinding (Anytime)



Strips Planning (Anytime vs AwA*)

		AwPBNF				AwBFPSDD				AwAPRA*			
		1.5	2	3	5	1.5	2	3	5	1.5	2	3	5
2 Threads	logistics-6	1.06	1.35	1.94	**1.98**	0.68	0.91	0.91	0.56	0.93	0.89	1.37	1.20
	blocks-14	1.91	1.99	13.22	22.4	1.02	1.18	7.71	11.9	1.33	8.67	**52.2**	6.65
	gripper-7	**2.05**	1.96	1.99	1.95	1.94	1.89	1.94	1.82	0.80	0.79	0.78	0.76
	satellite-6	1.58	1.96	**1.98**	1.91	1.85	1.87	1.49	1.80	0.75	0.79	0.79	0.80
	elevator-12	2.01	2.07	**2.13**	2.07	1.74	1.74	1.75	1.69	0.67	0.67	0.67	0.68
	freecell-3	1.93	1.06	2.78	**6.23**	1.45	1.46	1.97	3.08	1.30	1.32	4.71	1.44
	depots-7	1.94	2.00	2.01	**4.10**	1.44	1.45	1.32	2.40	1.22	1.29	1.26	2.69
	driverlog-11	1.95	**2.10**	1.99	0.77	1.73	1.78	1.59	1.41	1.16	1.20	1.14	1.21
	gripper-8	2.04	2.05	**2.09**	2.06	2.01	2.00	1.98	1.96	M	M	M	M
7 Threads	logistics-6	2.04	2.46	4.19	**4.21**	1.02	1.35	1.37	0.92	1.41	1.32	1.76	1.80
	blocks-14	3.72	22.4	25.7	7.20	1.60	1.96	12.1	19.9	2.49	15.2	99.2	**170**
	gripper-7	**5.61**	5.05	5.03	5.06	4.30	4.24	4.16	3.96	1.70	1.72	1.69	1.71
	satellite-6	**5.96**	4.66	5.74	4.70	4.10	3.54	4.16	3.88	1.27	1.27	1.28	1.29
	elevator-12	6.18	6.03	**6.20**	6.05	3.71	3.74	3.73	3.38	0.94	0.92	0.94	0.93
	freecell-3	3.54	1.50	**15.3**	11.5	1.78	1.82	2.59	4.14	3.57	3.71	11.8	4.28
	depots-7	5.74	5.52	5.48	**10.8**	2.02	1.96	1.92	3.68	M	M	M	M
	driverlog-11	5.78	**5.83**	5.73	2.18	2.58	2.86	2.57	2.34	M	M	M	M
	gripper-8	**5.82**	5.36	5.39	5.39	4.62	4.55	4.57	4.50	M	M	M	M

Summary

- Best-case performance better than serial in all domains tested.
- Higher weights sometimes hurt performance.

Conclusion

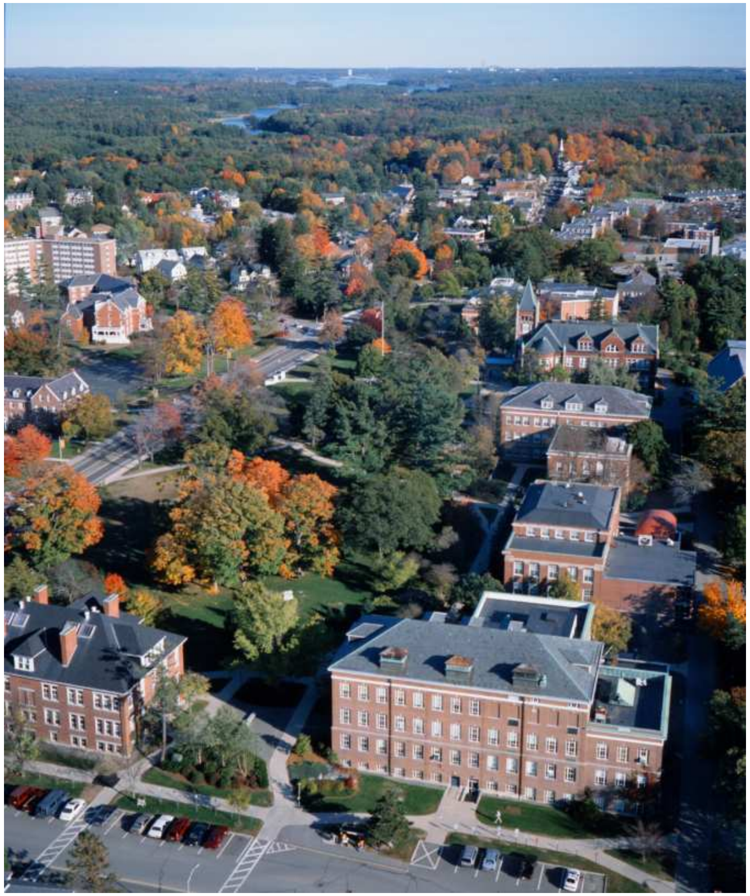
Parallel Best NBlock First:

- Fast
  - Beats all other algorithms used for comparison when returning optimal solutions.
  - More speedup for larger problems in suboptimal or anytime setting.
- Scales well
  - Tested out to eight threads.
- Easy to use
  - Only requires a user-provided abstraction.
- New pruning rules for suboptimal search.

For more information on optimal results, see IJCAI session on “Advances in A* Search” on Thursday, July 16 at 4PM.

The University of New Hampshire

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



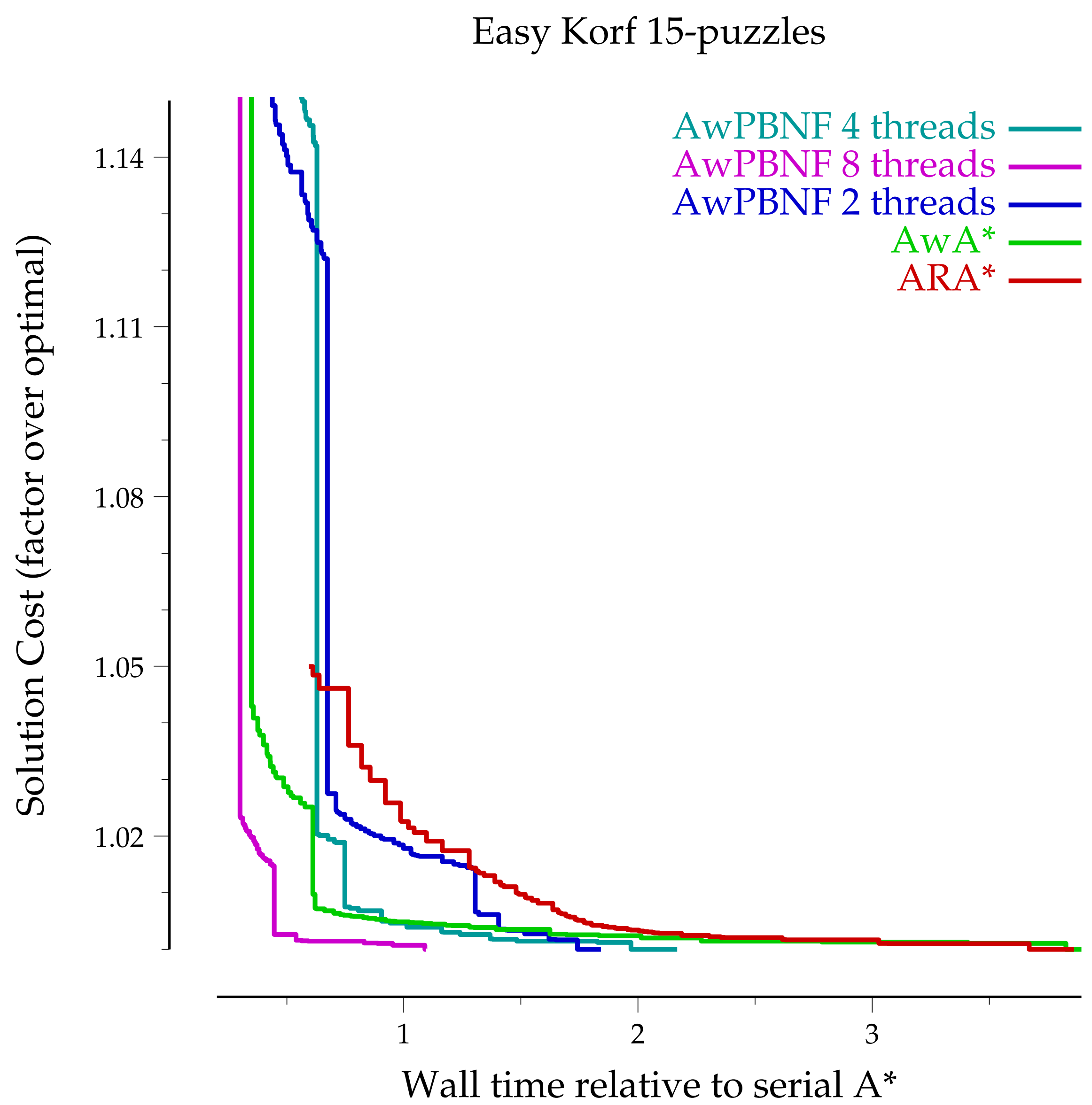
- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking, systems

Additional Slides

Four-way Grid Pathfinding (Bounded Suboptimal)

	wPBNF					wBFPSDD					wAPRA*				
w	1	2	4	5	8	1	2	4	5	8	1	2	4	5	8
1.1	0.83	1.49	2.69	3.14	**4.04**	0.87	1.31	1.97	2.21	2.57	0.93	1.50	2.09	1.99	1.76
1.2	0.76	1.38	2.44	2.77	**3.38**	0.83	1.24	1.84	2.06	2.22	0.88	1.40	1.92	1.83	1.58
1.4	0.52	0.99	1.60	1.75	**1.95**	0.78	1.15	1.57	1.68	1.53	0.55	0.92	1.26	1.22	1.05
1.8	0.53	0.61	0.67	0.68	0.65	0.72	**0.73**	0.67	0.63	0.47	0.51	0.68	0.63	0.51	0.44

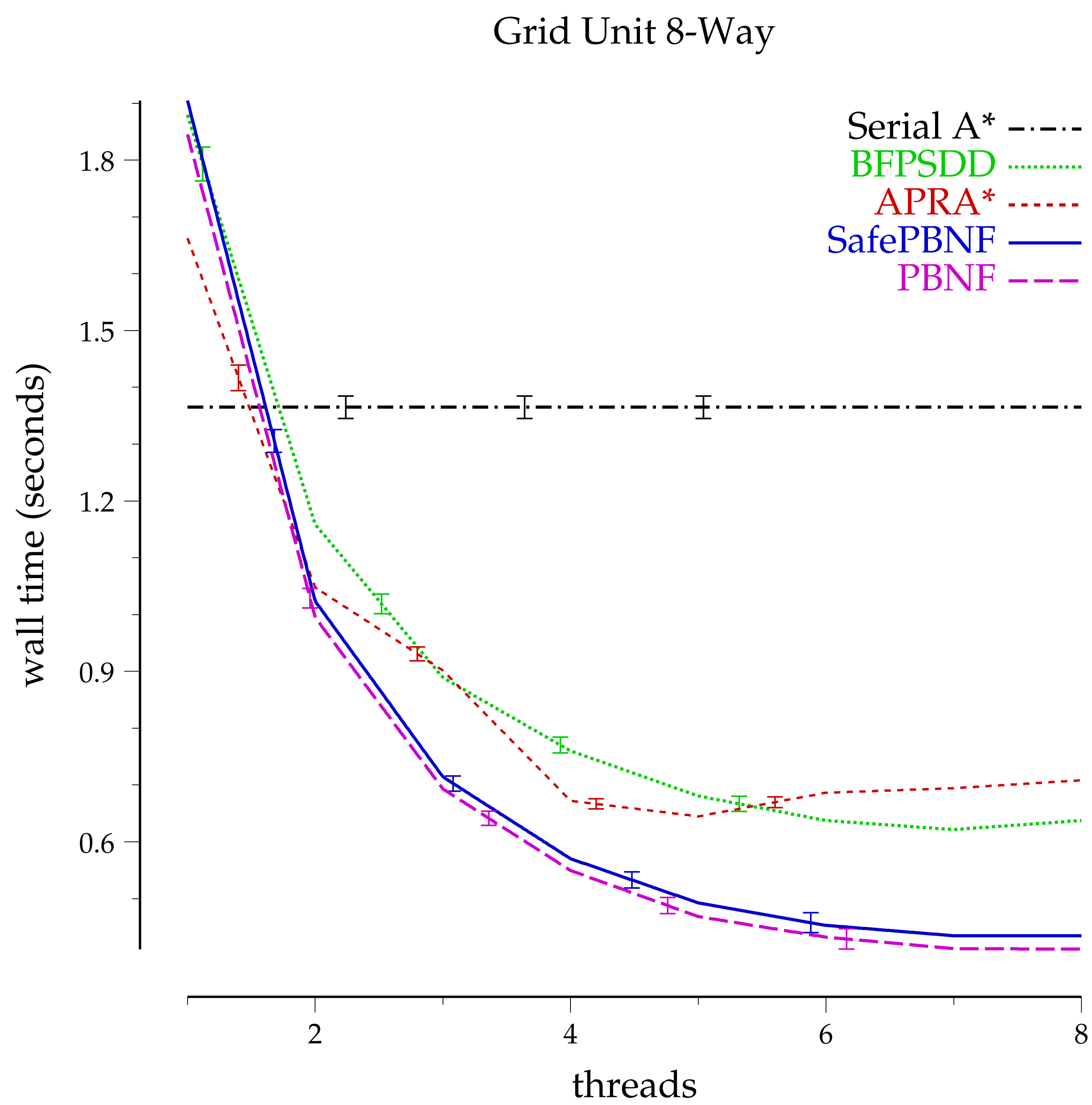
Easy Sliding 15-Puzzles (Anytime)



STRIPS Planning (Anytime vs PBNF)

7 Threads	AwPBNF				AwBFPSDD				AwAPRA*			
	1.5	2	3	5	1.5	2	3	5	1.5	2	3	5
logistics-6	1.48	1.84	**2.36**	2.27	0.68	0.93	0.71	0.54	1.12	1.08	1.08	0.98
blocks-14	1.24	1.22	0.21	0.03	0.87	0.18	0.16	0.16	**1.46**	1.46	1.42	0.94
gripper-7	**1.07**	0.99	0.99	1.00	0.93	0.95	0.93	0.92	0.99	1.03	1.01	0.99
satellite-6	**1.10**	0.87	1.08	0.88	0.88	0.77	0.91	0.90	0.99	1.00	1.01	1.02
elevator-12	**1.06**	1.04	1.04	1.03	0.77	0.78	0.76	0.73	1.02	1.00	1.00	1.00
freecell-3	1.05	0.44	0.99	0.29	0.64	0.64	0.20	0.14	1.13	**1.16**	0.82	0.10
depots-7	**1.20**	1.15	1.15	1.08	0.54	0.53	0.52	0.49	M	M	M	M
driverlog-11	1.16	1.15	**1.19**	0.43	0.53	0.58	0.54	0.50	M	M	M	M
gripper-8	**1.06**	0.99	0.99	1.00	0.99	0.98	0.99	0.97	M	M	M	M

Eight-way Grid Pathfinding (Optimal)



Eight-way Grid Pathfinding (Bounded Suboptimal)

	wPBNF					wBFPSDD					wAPRA*				
w	1	2	4	5	8	1	2	4	5	8	1	2	4	5	8
1.1	0.57	1.08	1.70	1.79	1.75	0.69	1.06	1.47	1.53	1.62	0.67	1.05	1.67	**1.87**	1.58
1.2	0.29	0.28	0.26	0.25	0.24	0.38	0.42	0.39	0.36	0.29	**0.78**	0.64	0.43	0.36	0.26
1.4	0.15	0.14	0.13	0.13	0.12	0.26	0.34	0.30	0.27	0.22	**0.77**	0.51	0.33	0.28	0.19
1.8	0.14	0.13	0.12	0.12	0.11	0.29	0.28	0.24	0.22	0.18	**0.77**	0.51	0.33	0.28	0.20

Eight-way Grid Pathfinding (Anytime)