Planning with Monte Carlo Random Walks: New Results

Martin Müller
University of Alberta
Edmonton, Canada

Joint work with:
Hootan Nakhost, Fan Xie, Richard Valenzano
(U Alberta)

Contents

- Introduction - classical planning
- Planning with Monte Carlo Random Walks
- Recent progress: smart restarts, local tree search (LTS)
- The Arvand and Arvand Herd planners
- The 2011 International Planning Competition (IPC)

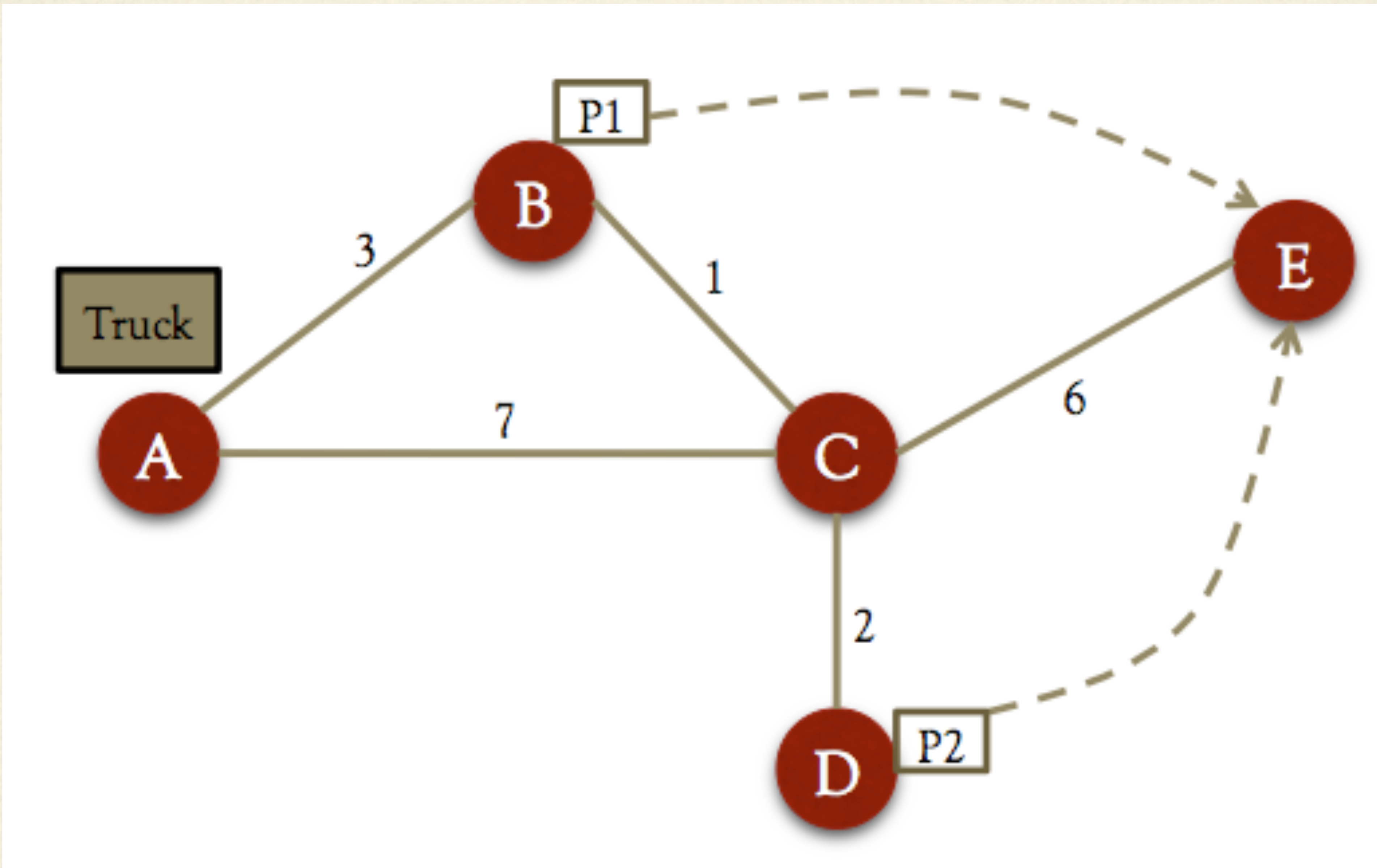
Planning

- Planning - given a *problem*, find a *plan* to solve it
- Very many variations, such as:
  - path planning in robotics
  - adversarial planning in games
  - planning with resource constraints - time, fuel,...

Classical Planning - Graph Search

- **States** represented by boolean predicates or by multi-valued variables
- **Actions** defined by **preconditions** and **effects**
- **Initial state** S
- **Goal** conditions G - subset of state
- **Plan** = any path from S to state satisfying G
- deterministic, complete information

Example: Transportation



Planning Community

- Well-established community
- ICAPS conference, sessions at IJCAI, AAAI, ECAI, SOCS,...
- Planning competitions - IPC 2011

Approaches to Planning

- Forward search is most popular
- Translations to SAT work well too (Kautz, Rintanen)
- Strong, slow evaluation functions
  - Relaxed planning graph (Hoffmann)
  - Landmark heuristics

Search-based Planning

- Used in most strong current planners
- Heuristic function h to evaluate distance to a goal state
  - Example: FF heuristic, h_{FF}
- Mostly greedy search, e.g. hill-climbing, weighted A* (WA*), greedy best-first search (gbfs)

Problems

- Strong planning heuristics are very slow
- Based on solving relaxed problem - ignore "negative effects"
- Most planners use greedy searches:
  - Almost all *exploitation*
  - Lack of *exploration*

The Arvand Planner

- Nakhost & Müller, IJCAI 2009
- Idea: apply lessons from games research to planning
- Background: work on Monte Carlo algorithms (UCB, UCT, MCTS) shows importance of exploration in search
- Breakthrough performance in Go, many other games. Nested MC search (Cazenave)

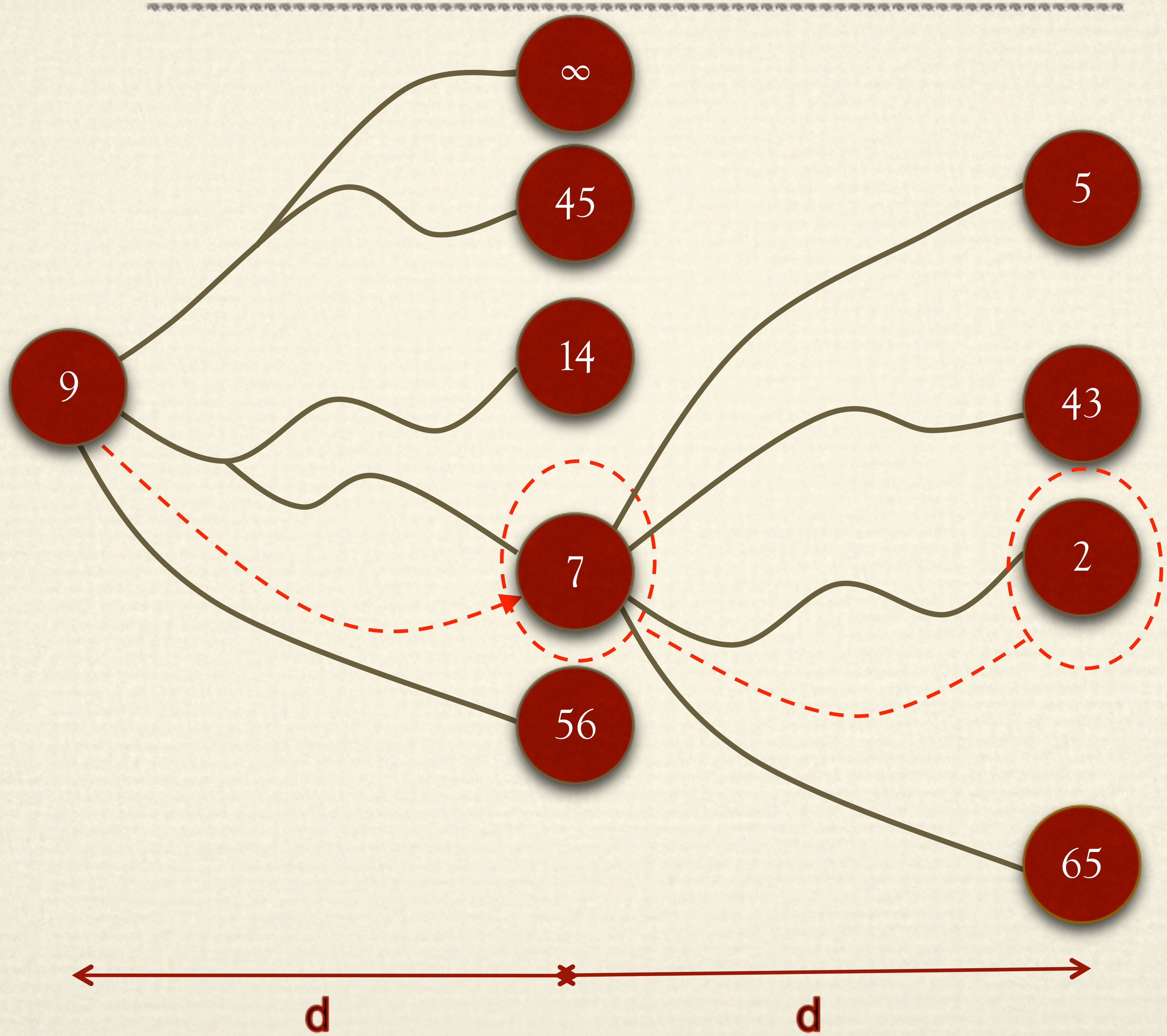
Exploration in Planning

- Still use a heuristic *h*, but not at every step
  - Generating set of all legal actions is fast
  - Evaluation is 2 orders of magnitude slower
- Explore local *neighbourhood* of state before choosing next actions
- Simplest way of exploration: random actions

Monte Carlo Random Walks

- Follow random sequence of actions for *d* (e.g. *d=10*) steps, then evaluate endpoint
- Repeat many times (e.g. *n=2000*)
- *End point search continuation*: Jump to best encountered endpoint
- If no improvement, restart

Random Walks



Advantages

- Deals directly with issue of exploration, randomization
- Can escape quickly from local minima, plateaus
- Exploits greater speed of action generation
- Simple planner, surprisingly powerful
  - Good in coverage - number of problems solved
  - Good scaling to larger problem instances

Disadvantages

- Poor plan quality - plan consists of concatenated random sequences

  → use plan improvement postprocessor

- Not systematic - may miss the only good action

  → use portfolio planner

- Slower on easy problems, where exploration is not needed

  (→ use portfolio planner)

Other Differences - Good or Bad?

- Randomization
  - No guarantee to find solution
  - Can escape from traps where deterministic algorithm gets stuck
- Low memory usage
  - Can run forever, while e.g. WA* quickly exhausts memory
  - On restart, can not profit from previous good runs

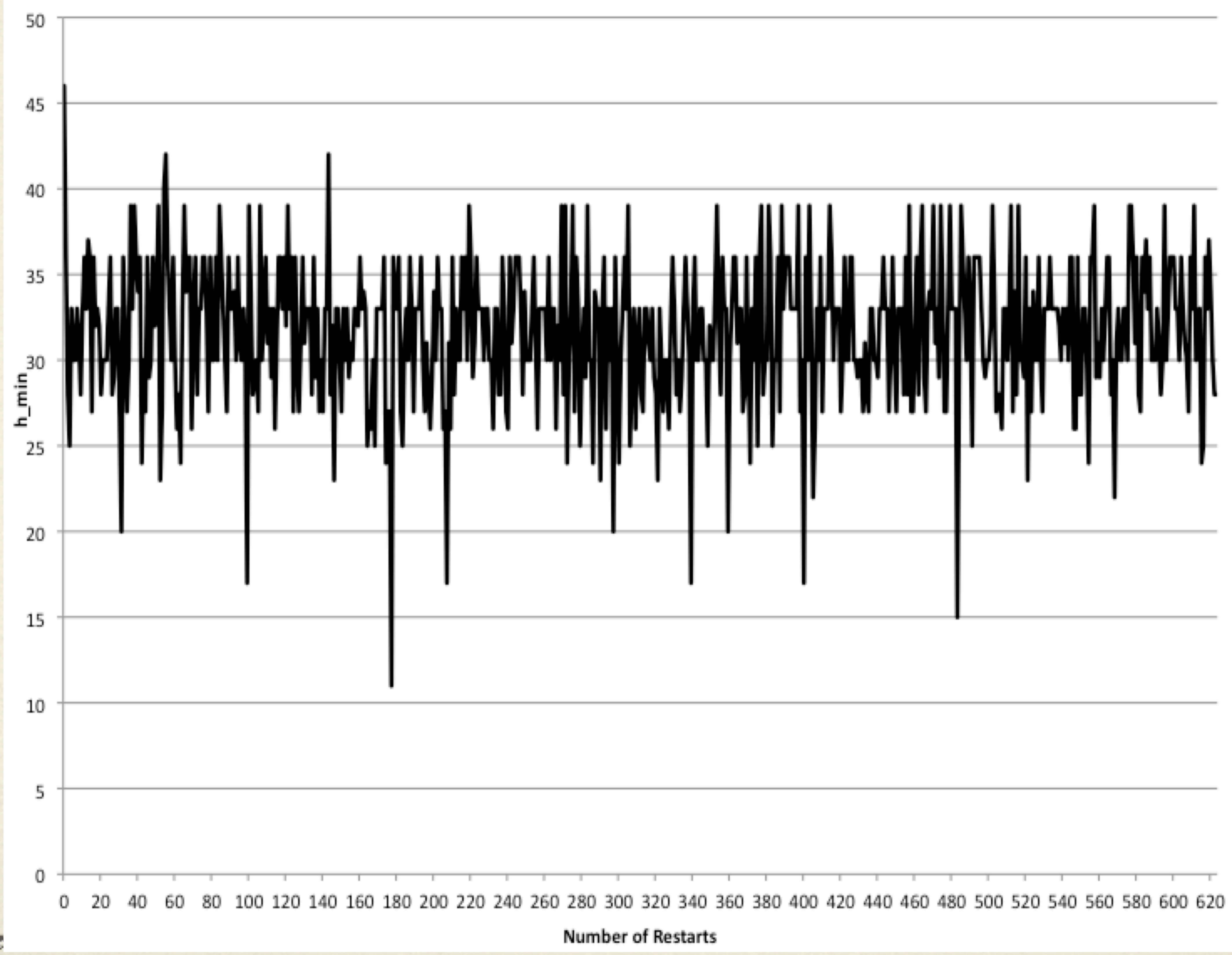
Improvements

- Modify length and number of random walks
- MDA - Try to avoid *deadlock* states
- MHA - Prefer *helpful* actions
- Smart restarts - re-use pool of previous good plan fragments
- Use local tree search
- Use portfolio with other types of planners

Smart Restarts

- Arvand's previous strategy: basic restarts
  - forward chaining local search
  - In each step, use MRW to find next state
  - If no progress, restart from beginning

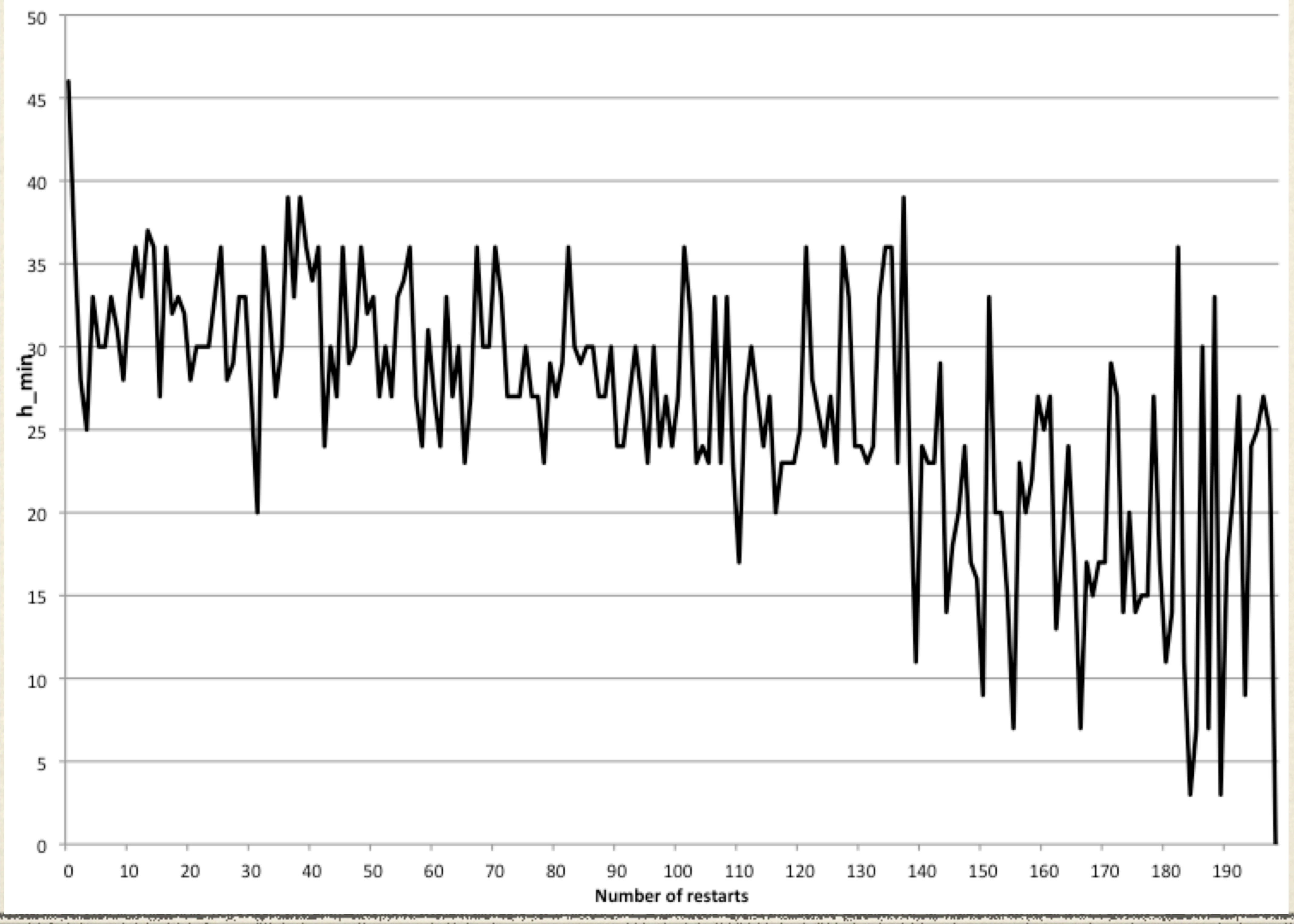
Basic Restarts - best h values



Smart Restarts

- Keep pool of most promising search episodes so far
- Restarting from random state of random episode in pool
- Main parameters:
  - pool size p
  - replacement policy

Smart Restarts - best h values

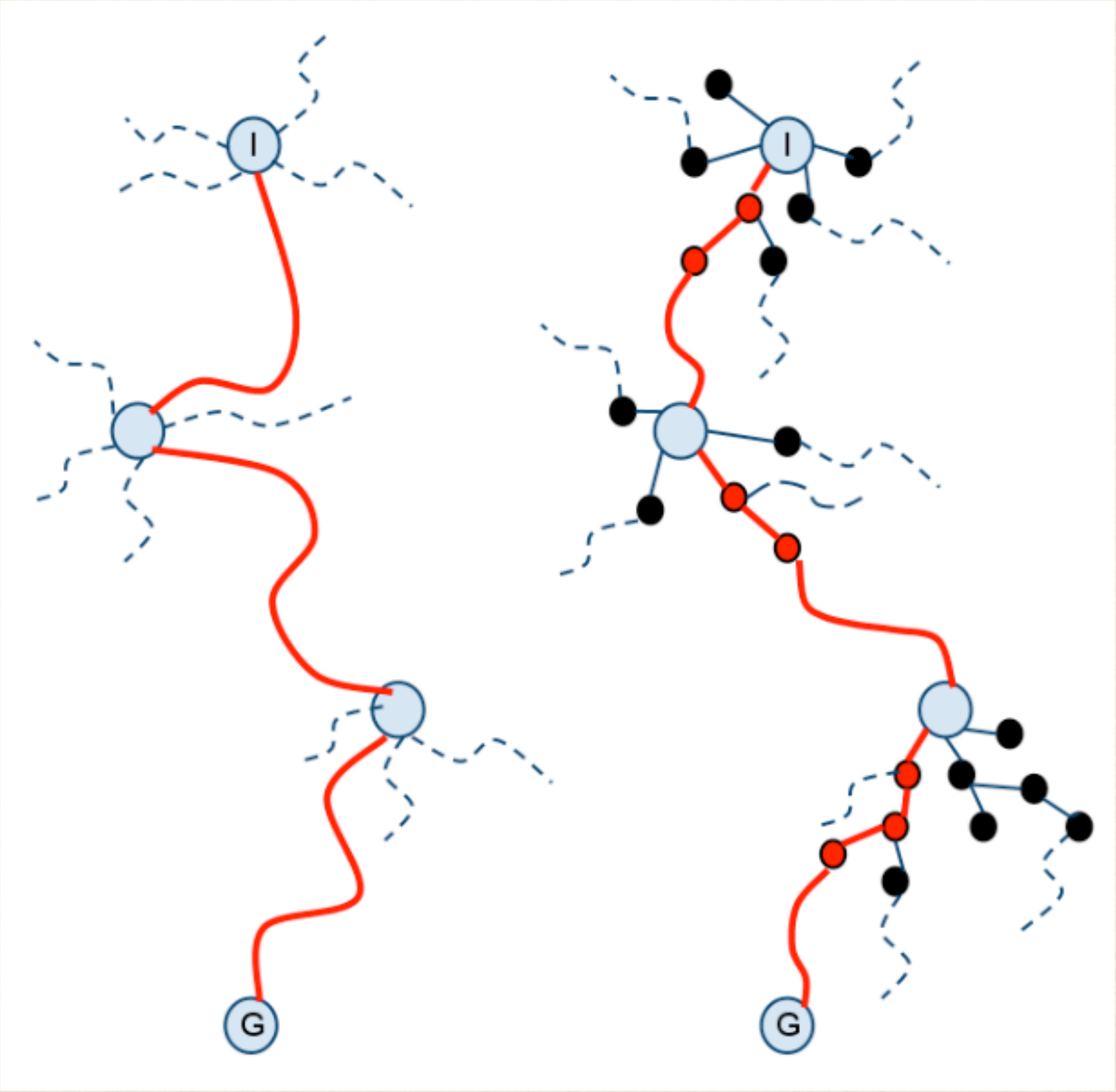


Local Tree Search (Xie et al 2011)

- Combine tree search with random walks
- More systematic search before each jump
- Tree growth:
  - epsilon-greedy child selection
  - run random walk probe from leaf node
  - evaluate nodes by best probe in subtree
  - After *n* steps, jump greedily

Illustration - Arvand vs LTS

Arvand



LTS

Example

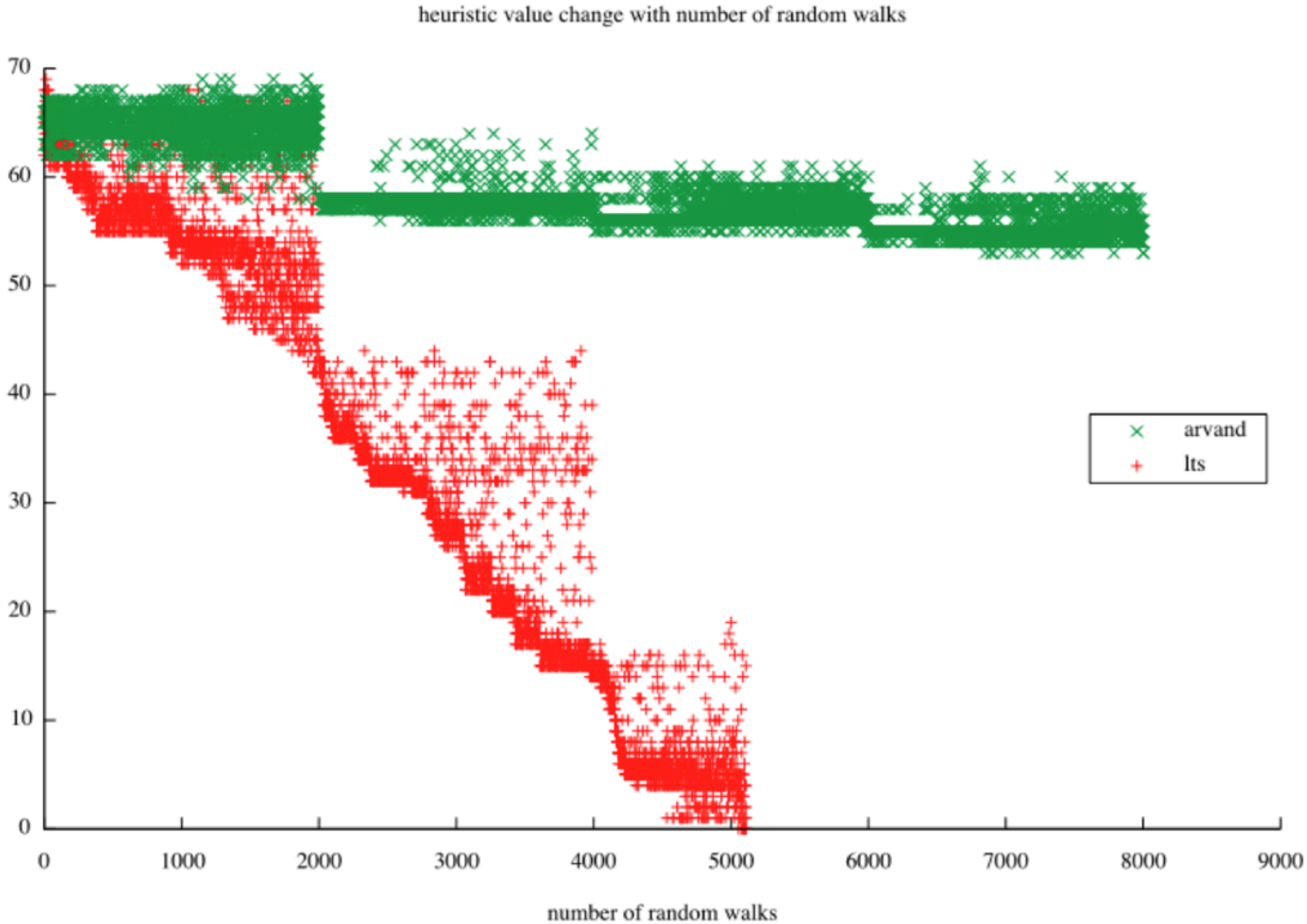
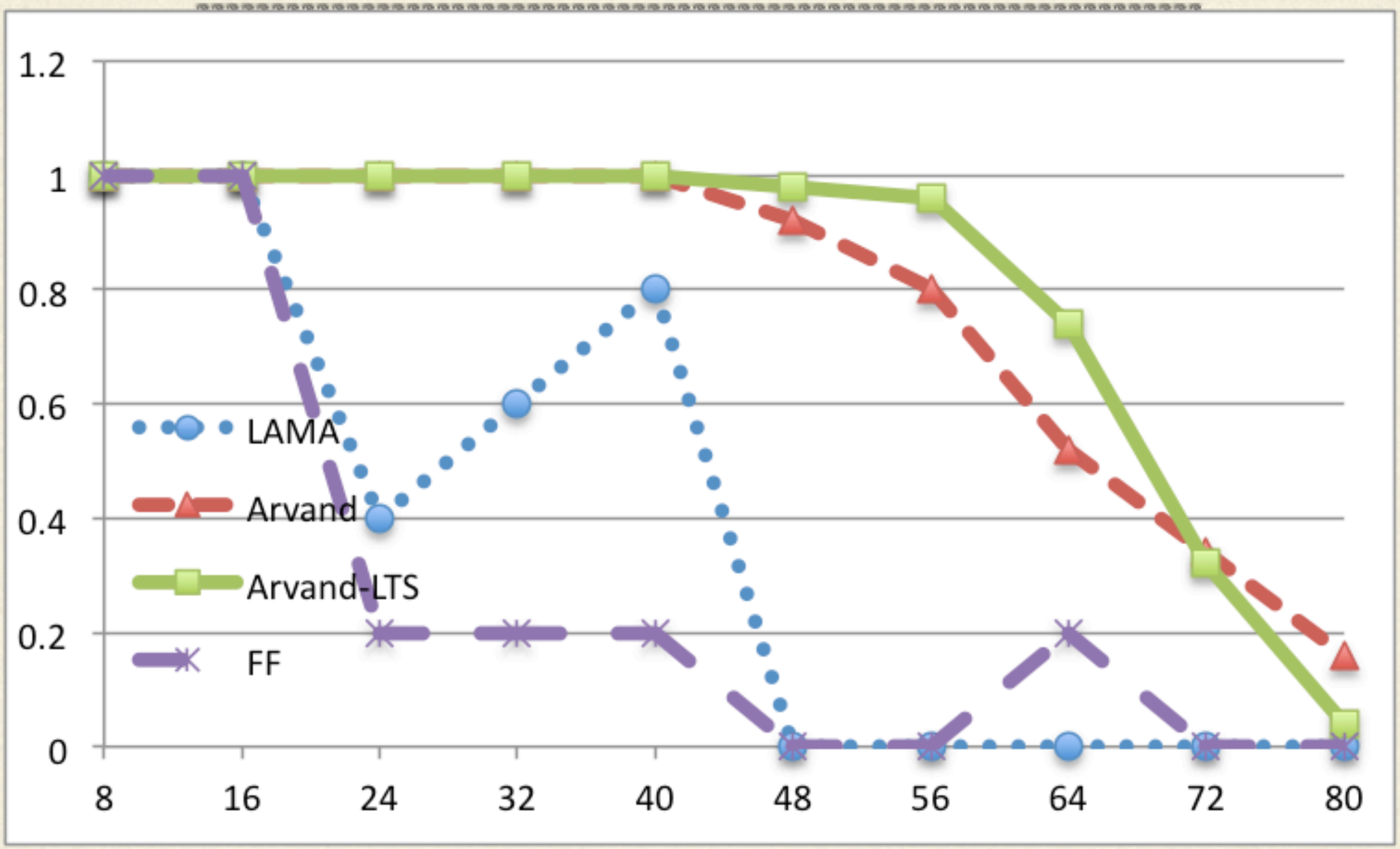


Figure 2: h-value change in elevator-16

Scaling to Larger Problems



- Example: woodworking domain

Aras Postprocessor

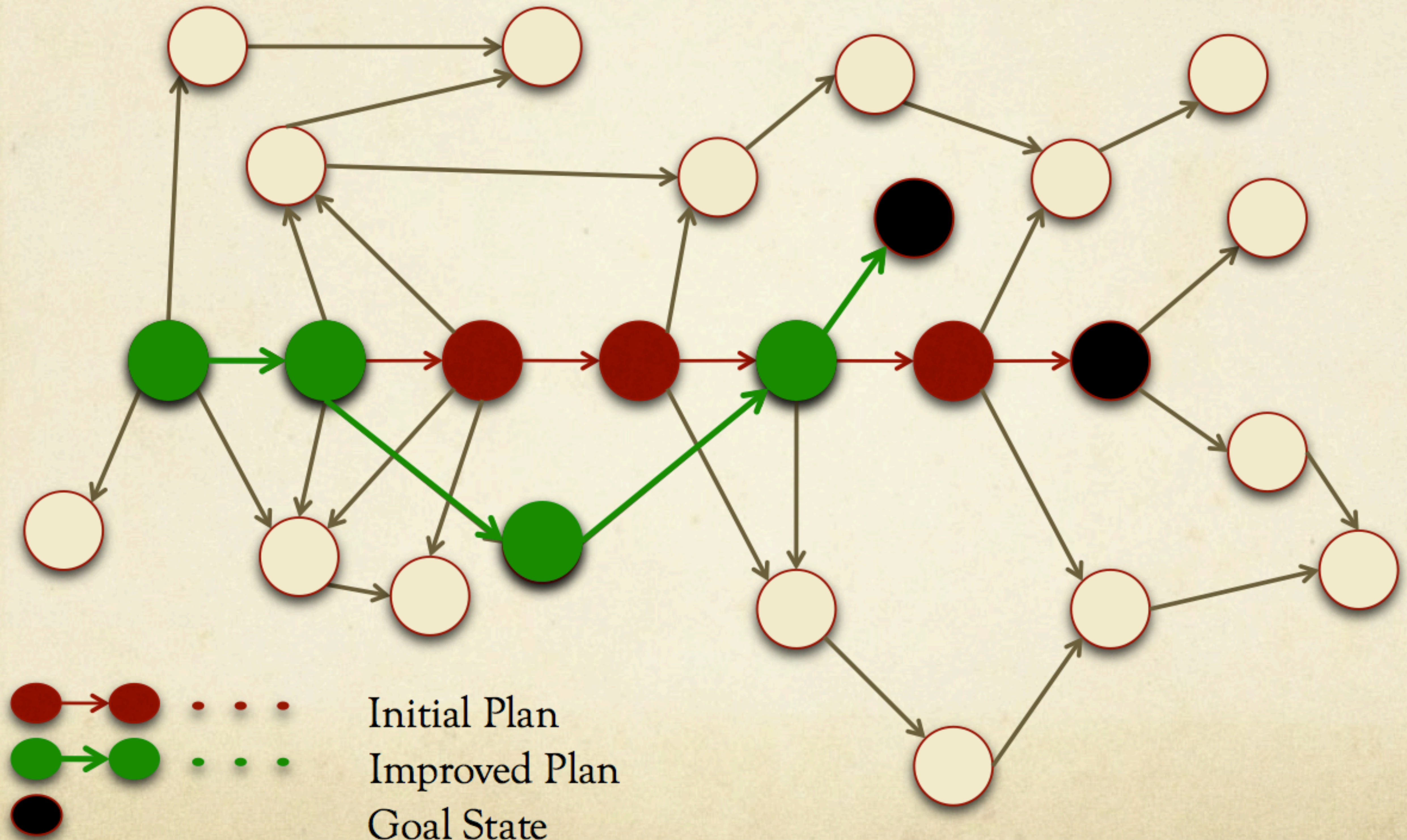
- Nakhost and Müller, ICAPS 2010
- Problem of Arvand: low plan quality
- Aras Postprocessor: improve given plan
- Two main techniques:
  - Action elimination
  - Plan neighbourhood graph search

Action Elimination

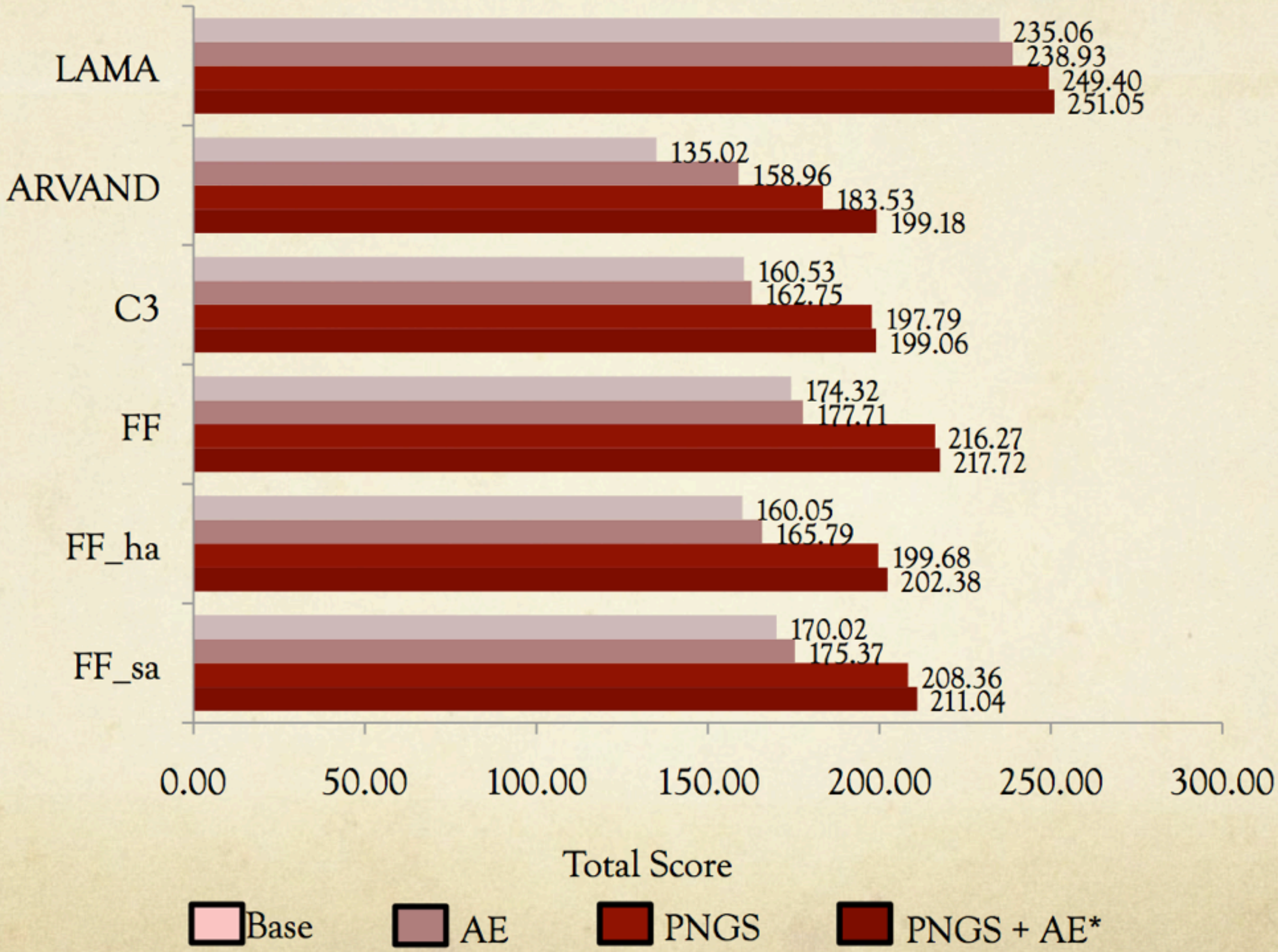
- Try to remove unnecessary actions from plan
- Try to remove any one action
- Remove every other action that loses support
- Check if result is still valid plan
- If not, undo changes

Plan Neighborhood Graph Search

- Start with valid plan
- Build neighbourhood of states near every state along the plan trajectory
- Find shortest path in this graph
- While (not out of resources)
  - Extend size of neighborhood
  - Repeat


Initial Plan
Improved Plan
Goal State

IPC-2008 PNGS and AE



Parallel System: Arvand Herd

- Anytime multicore planner developed for IPC
  - 4 core version:
  - 3 copies of Arvand w. randomized parameters
  - 1 copy of LAMA2008 (winner of 2008 IPC)
  - Shared restart pool
  - Aras postprocessor run on all solutions

7th International Planning Competition (IPC) 2011

- Organized by Ángel García-Olaya, Sergio Jiménez, Carlos Linares López, Universidad Carlos III de Madrid
  - Thanks for tables and graphic of results!!!
- Previous IPC: 1998, 2000, 2002, 2004, 2006, 2008
- *http://ipc.icaps-conference.org/*

Basic Rules

- Submit planners as source code
- Can compile, test on competition systems
- Only “trivial” bug fixes allowed later
- “Blind” evaluation: domains not known before or during the contest
  - Published now, after the competition

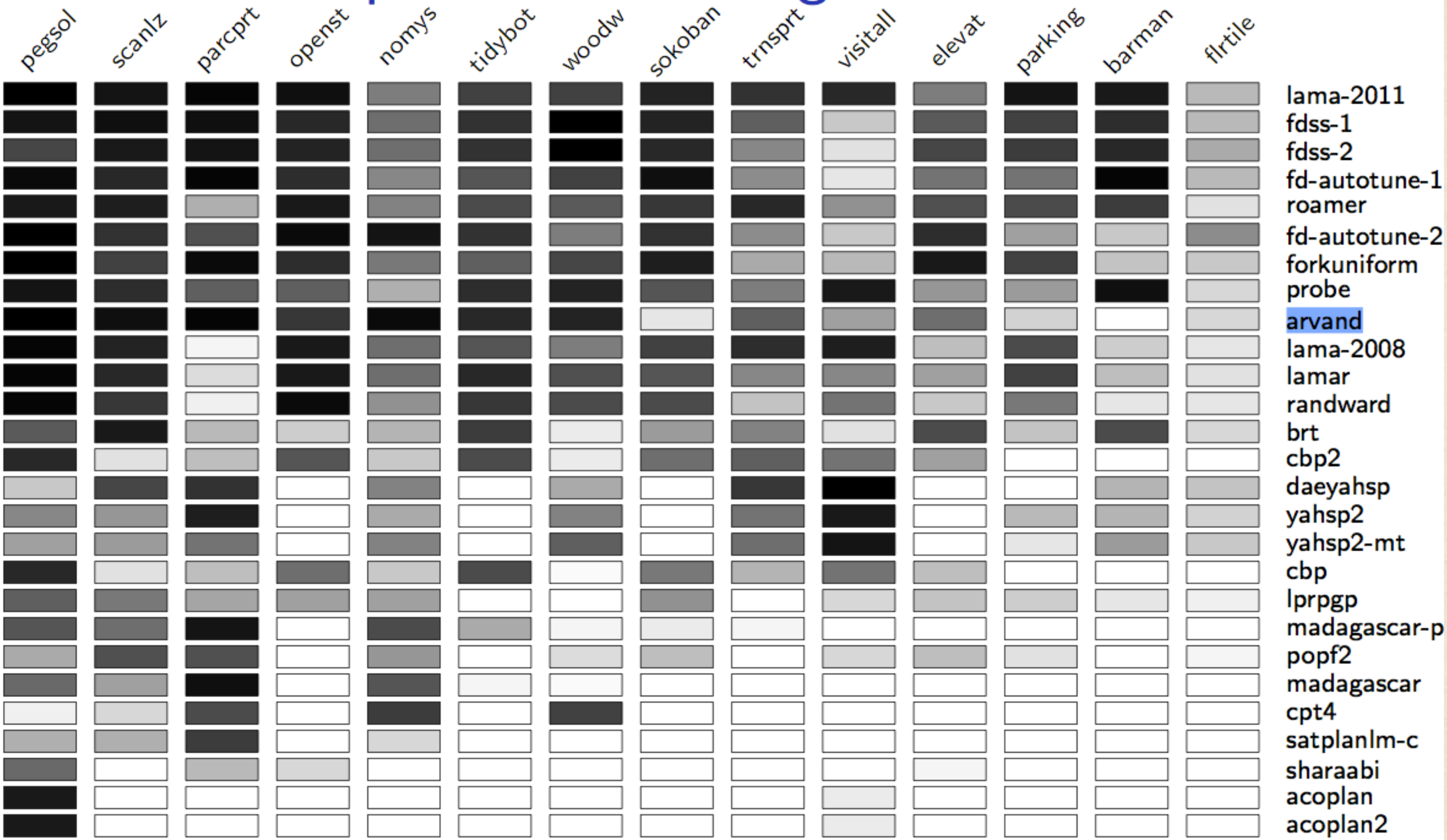
Rules, Satisficing Tracks

- 14 planning domains, 20 instances per domain = 280 problems in total
  - 9 old, 5 new domains
  - In old domains, usually 10 previous and 10 new, harder instances
- 30 minutes per instance, 6 Gb memory

Deterministic Tracks

- "Sequential satisficing": single-core
  - 27 participants, including *Arvand*
  - Previous IPC winner: LAMA2008
- "Sequential multicore satisficing": 4 cores, shared memory
  - 8 participants, including *Arvand Herd*
  - First multicore competition

Sequential Satisficing track: Results



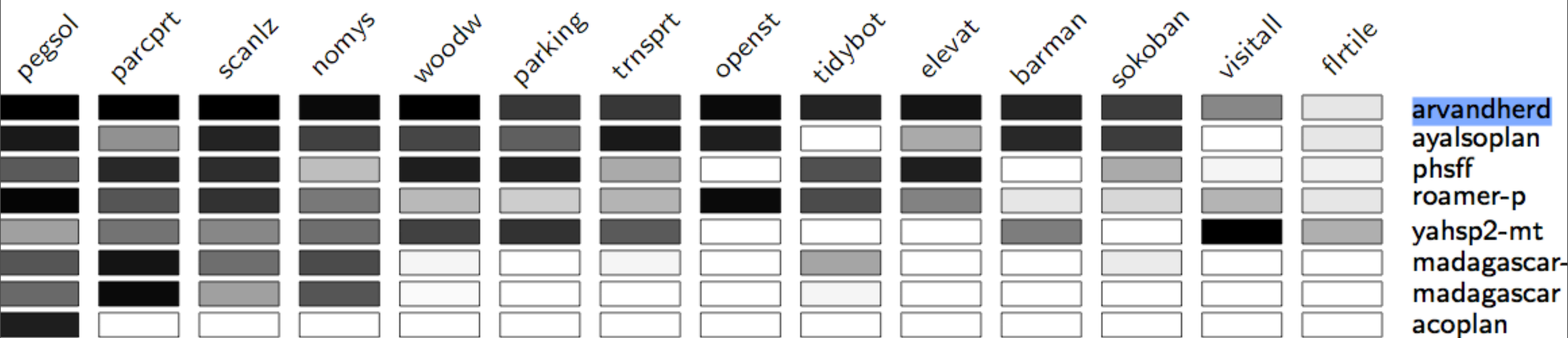
Single Core Results

planner	pegsol	scanalyzer	parcprinter	openstacks	nomystery	tidybot	woodworking	sokoban	transport	visitall	elevators	parking	barman	floortile	total
lama-2011	20.00	18.12	19.30	18.58	9.92	14.71	14.64	17.22	15.74	16.51	10.28	18.11	17.70	5.49	**216.33**
fdss-1	18.49	18.52	18.68	16.86	11.26	16.09	19.99	17.05	12.22	3.97	12.52	14.79	16.34	5.30	**202.08**
fdss-2	14.44	17.86	18.31	16.94	11.21	16.08	19.82	16.67	9.37	2.12	14.50	15.28	16.81	6.60	**196.00**
fd-autotune-1	19.23	16.67	19.40	16.28	9.50	13.25	14.71	18.57	8.99	1.71	11.04	10.93	19.37	5.46	**185.09**
roamer	17.74	17.59	6.22	17.80	9.67	13.99	12.51	15.35	16.72	8.62	13.61	14.08	15.18	2.38	**181.47**
fd-autotune-2	19.95	15.95	13.57	19.09	18.36	15.89	10.24	15.93	8.79	4.14	16.17	7.19	4.01	8.87	**178.15**
forkuniform	19.90	14.88	19.28	16.22	10.45	12.44	14.49	17.35	6.61	5.19	18.01	14.59	4.47	4.02	**177.91**
probe	18.44	16.34	12.11	12.41	5.90	16.32	17.10	13.14	10.03	18.05	8.24	7.63	18.60	2.83	**177.14**
arvand	20.00	18.53	19.42	15.38	18.97	16.56	17.05	2.00	12.25	7.38	11.22	3.31	0.00	3.00	**165.07**
lama-2008	19.54	17.15	0.88	18.05	11.44	13.17	9.97	14.62	16.44	17.59	4.94	13.86	3.60	2.07	**163.33**
lamar	19.36	16.71	2.55	17.96	11.46	16.67	13.63	12.99	9.17	9.17	7.34	14.76	5.08	2.36	**159.20**
randward	19.58	15.68	1.00	18.93	8.55	15.57	13.83	14.01	4.46	10.92	4.29	10.55	2.06	2.00	**141.43**
brt	12.68	17.77	5.17	3.75	5.75	14.91	1.64	7.66	9.65	2.13	13.84	4.42	13.83	2.82	**116.01**
cbp2	16.64	2.29	5.00	13.29	4.00	13.78	1.63	11.39	12.16	10.82	7.34	0.00	0.00	0.00	**98.34**
daeyahsp	4.00	14.23	15.70	0.00	9.67	0.00	6.32	0.00	15.48	19.71	0.00	0.00	5.72	4.39	**95.23**
yahsp2	9.46	8.08	17.70	0.00	6.70	0.00	9.65	0.00	10.92	18.09	0.00	5.24	5.85	3.29	**94.97**
yahsp2-mt	7.43	7.63	10.95	0.00	9.61	0.00	12.41	0.00	11.39	18.18	0.00	1.73	7.55	4.08	**90.95**
cbp	16.58	2.29	5.00	11.30	4.00	13.83	0.49	10.55	5.73	10.82	4.86	0.00	0.00	0.00	**85.43**
lprpgp	12.43	10.63	6.86	7.21	7.26	0.00	0.00	8.51	0.00	2.82	4.56	3.89	1.81	1.09	**67.07**
madagascar-p	12.98	11.30	18.31	0.00	13.93	6.60	0.71	1.50	0.60	0.00	0.00	0.00	0.00	0.00	**65.93**
popf2	6.39	13.65	13.58	0.00	8.22	0.00	2.51	4.92	0.00	2.82	4.73	2.40	0.00	0.67	**59.88**
madagascar	11.83	7.11	18.88	0.00	12.98	0.73	0.45	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**51.98**
cpt4	1.00	3.00	14.00	0.00	15.00	0.00	14.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**47.85**
satplanlm-c	6.00	5.92	15.04	0.00	3.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**29.96**
sharaabi	11.88	0.00	5.26	2.81	0.00	0.00	0.00	0.00	0.00	0.00	0.56	0.00	0.00	0.00	**20.52**
acoplan	17.92	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.41	0.00	0.00	0.00	0.00	**19.33**
acoplan2	17.75	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.34	0.00	0.00	0.00	0.00	**19.09**
total	**391.63**	**307.90**	**302.18**	**242.85**	**236.79**	**230.58**	**228.65**	**219.43**	**196.71**	**193.49**	**168.04**	**162.78**	**157.98**	**66.73**	
	++	++	++	+	++	+	+	--	*	-	*	--	--	--	

Discussion - Single Core

- Arvand in 9th place out of 27 planners
- Strong performance in “easy” domains, but very weak in “hard” domains
  - Best in 4 of 14 domains, close to best in 3 more
  - Terrible in 4-5 puzzle-like domains, score close to 0
    - These domains favour a more systematic search

Sequential Multi-core track: Results



Multicore Results

planner	pegsol	parcprinter	scanalyzer	nomystery	woodworking	parking	transport	openstacks	tidybot	elevators	barman	sokoban	visitall	floortile	total
arvandherd	20.00	19.80	19.74	19.00	20.00	15.34	15.47	18.98	17.25	18.10	17.00	15.00	9.38	2.00	**227.07**
ayalsoplan	17.87	8.56	16.92	14.50	14.10	12.46	18.01	17.35	0.00	6.57	16.83	14.93	0.00	1.85	**159.95**
phsff	12.62	16.53	16.36	4.85	17.48	17.14	6.33	0.00	13.65	17.33	0.00	6.65	0.61	1.04	**130.59**
roamer-p	19.60	13.10	16.02	10.61	5.15	3.70	5.78	18.97	13.94	9.65	1.92	3.00	5.74	1.88	**129.06**
yahsp2-mt	7.34	11.00	9.15	11.42	14.76	16.08	12.80	0.00	0.00	0.00	10.12	0.00	19.84	6.08	**118.58**
madagascar-p	12.98	18.34	11.45	13.93	0.71	0.00	0.69	0.00	6.84	0.00	0.00	1.50	0.00	0.00	**66.44**
madagascar	11.83	18.90	7.11	12.98	0.45	0.00	0.00	0.00	0.73	0.00	0.00	0.00	0.00	0.00	**52.00**
acoplan	17.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**17.62**
total	**119.85**	**106.25**	**96.73**	**87.28**	**72.66**	**64.71**	**59.09**	**55.30**	**52.41**	**51.65**	**45.87**	**41.09**	**35.58**	**12.85**	

- *Arvand Herd* won by a large margin!
- Consistently strong results over almost all domains
- The *LAMA* component covers puzzle-like domains
- Synergy in the elevators, barman domains?

Single core Winner: LAMA 2011

- by Silvia Richter (NICTA), Matthias Westphal, Malte Helmert (Univ. Freiburg)
- Re-implementation of previous winner LAMA2008 in the current Fast Downward framework
- Main change (as in many other top planners...)
    - run greedy best-first search first
    - *ignore* action costs in this run
    - increases coverage

Some Other Planners at IPC

- *Probe*:
  - greedy best first search, plus run a single high-quality *probe* from each state
  - More than half of previous IPC problems solved by single probe, without search!
- *Fast Downward Stone Soup*:
  - simple portfolio planner, tuned on previous IPC

Other Planners (2)

- *Fast Downward Autotune*
  - Uses stochastic parameter optimization system *ParamILS*
  - 2000 training instances, including previous IPC
- *Roamer*
  - Combine best-first search with random walks to escape from plateaus, local minima

Lesson Learned, Future Work

- Move away from focus on heuristics, more focus on search
- Portfolio and multi-queue search methods are here to stay
  - Try *Arvand* + *LAMA2011* + *Probe* + *Aras*
- Analyze components of success of *Arvand Herd*
  - *Arvand*, *LAMA2008*, *Aras* postprocessor
  - Try *Arvand Herd* in single core track

Summary

- Monte-Carlo random walks in planning
- Basic idea is already quite strong. Many refinements
- Good scaling to larger problems
- Does not work with puzzle-like domains
- Strong results at IPC with portfolio system *Arvand Herd*