Linköping University

Automated Planning

Performance:
Planning competitions +
Combining and tuning planners

Jonas Kvarnström
Department of Computer and Information Science
Linköping University

jonas.kvarnstrom@liu.se – 2019

Planning Competitions

International Planning Competitions

jonkv@ida

- Started in 1998
  - Origin of the Planning Domain Definition Language
- Held irregularly, with different tracks
  - Rules have varied somewhat over time

Sequential Track Rules

jonkv@ida

- Current rules (approximately):
  - **Sequential Satisficing Track:**
    - For each problem instance, spend up to 30 minutes searching
    - Return the highest quality (lowest cost) solution you found – *if* you found one
    - As long as you found it in 30 minutes, speed is irrelevant
  - **Sequential Optimal Track:**
    - For each problem instance, spend up to 30 minutes searching
    - Return an optimal solution or none at all
    - Suboptimal in *one* domain ➔ score 0 for that domain
    - Suboptimal in *multiple* domains ➔ disqualified
  - **Sequential Agile Track:**
    - For each problem instance, spend up to 5 minutes searching
    - Only speed counts: Return the first solution you find

IPC Scores: Satisficing

jonkv@ida

- Scores for satisficing planning:
  - Planner C, problem instance p
  - $$\text{score}(C, p) = \begin{cases} 0 & \text{if planner } C \text{ did not solve } p \\ \dfrac{\text{cost of best solution}}{\text{cost of solution found by } C} & \text{if planner } C \text{ did solve } p \end{cases}$$
  - If the best cost *found by any planner* is 30, and you found a plan of cost 40, your score is $30/40 = 0.75$
  - Total score = sum of all individual problem instance scores

IPC Scores: Optimizing

jonkv@ida

- Scores for optimizing planning:
  - Planner C, problem instance p
  - $\text{score}(C, p) = \begin{cases} 0 & \text{if planner } C \text{ did not solve } p \\ 1 & \text{if planner } C \text{ did solve } p \end{cases}$

Our first results: 2008

IPC 2008

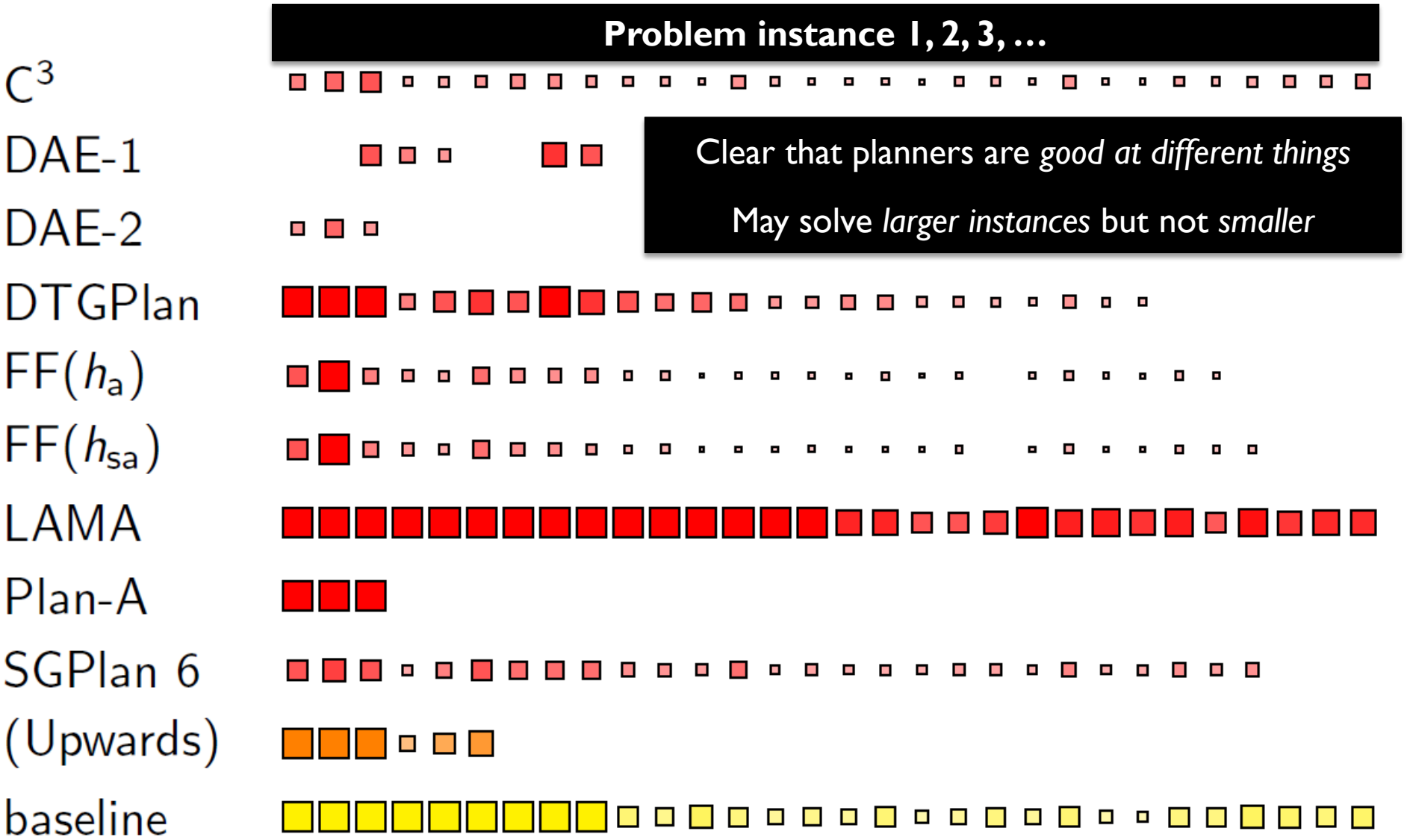
jonkv@ida

- International Planning Competition 2008:
  - First time that the domains were ***secret***
  - First time that the experiments were ***run by the organizers***
  - First time the performance scores were ***clearly defined in advance***

Sequential Satisficing (1)

jonkv@ida

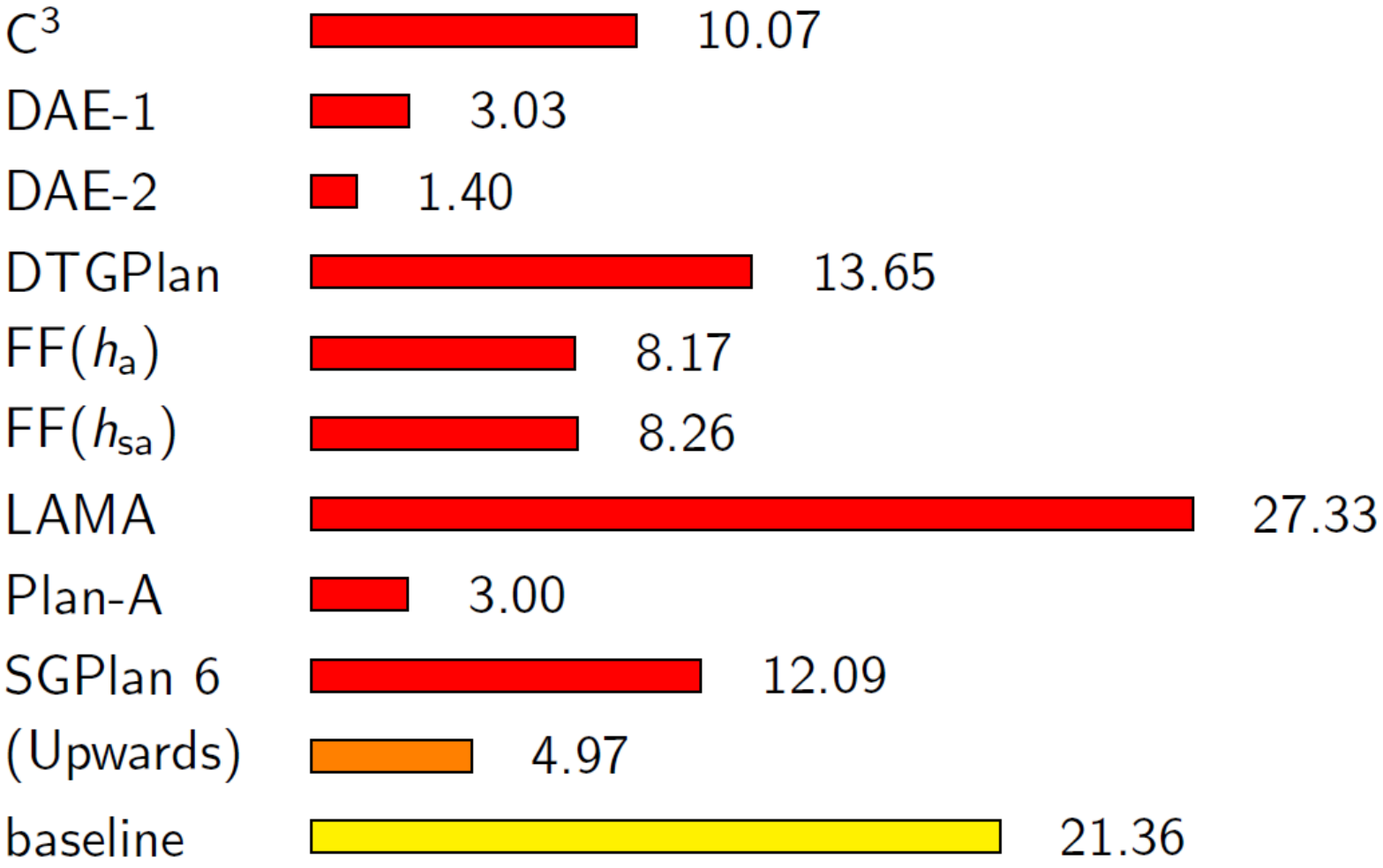
- Example performance *per problem instance*: OpenStacks domain



Sequential Satisficing (2)

jonkv@ida

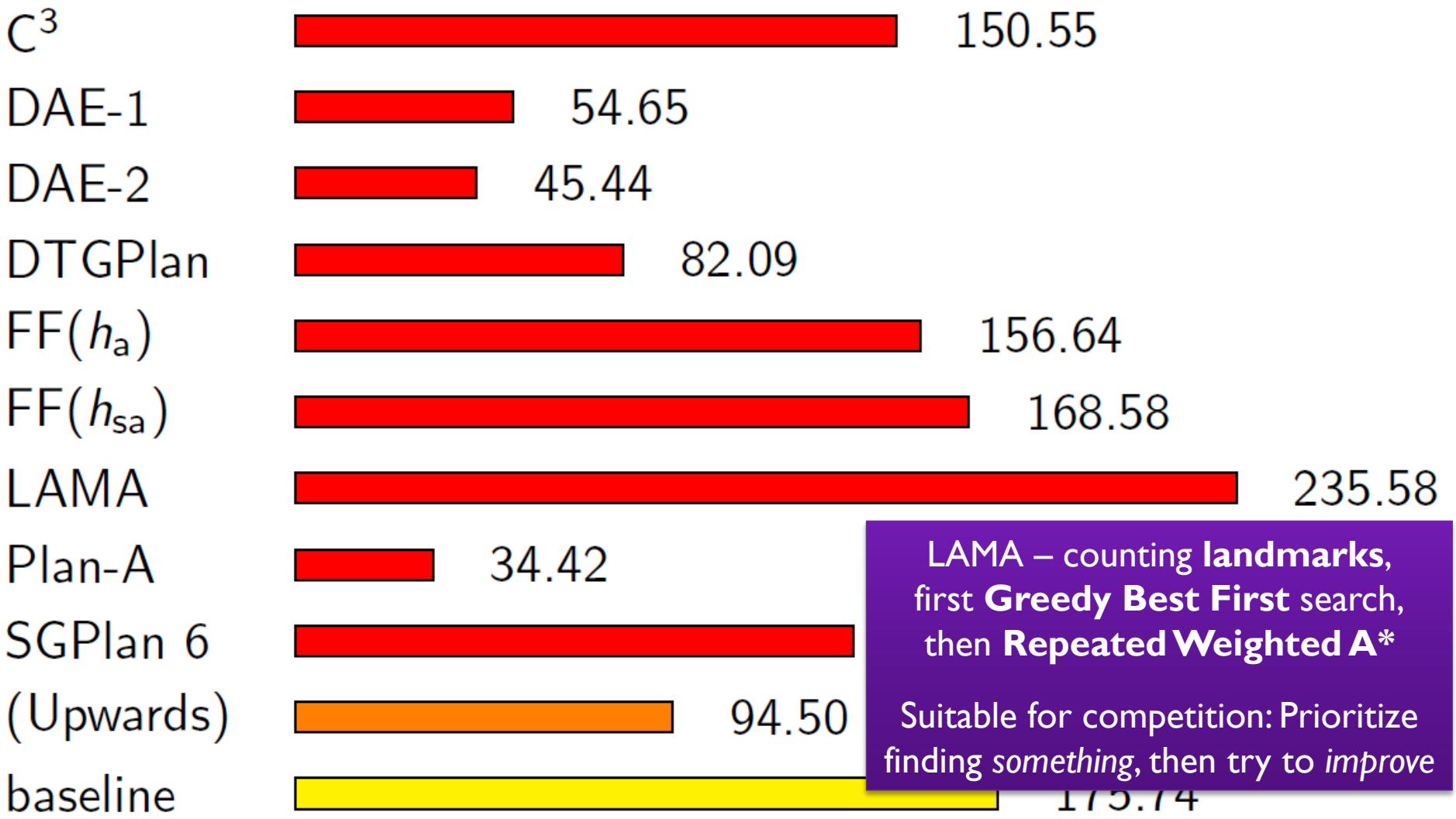
- Example performance *summarized for one domain*



Sequential Satisficing (3)

jonkv@ida

- Total performance in **sequential satisficing**



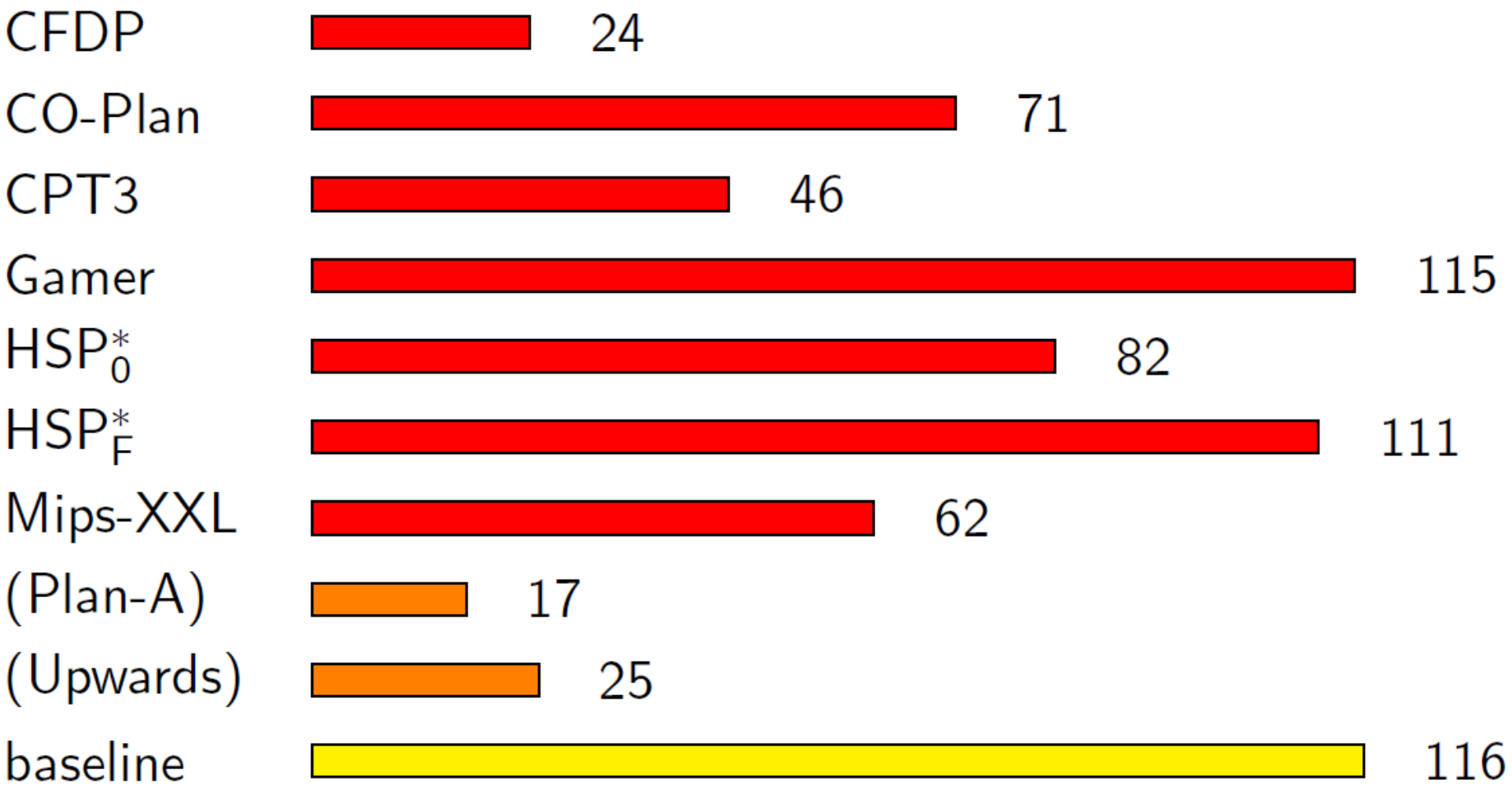
LAMA – counting **landmarks**, first **Greedy Best First** search, then **Repeated Weighted A***

Suitable for competition: Prioritize finding *something*, then try to *improve*

Sequential Optimizing

jonkv@ida

- Total performance in **sequential optimizing**



Parameter Optimization

Parameter Optimization (1)

jonkv@ida

- Some planners have **many parameters** to tweak
  - In early planning competitions, domains were **known in advance**
    - Participants could manually adapt their ”domain-independent” planners…
  - Somewhat **exaggerated quote** from IPC-2008 results:
    - if domain name begins with “PS” and part after first letter is “SR”:
      use algorithm 100
    - else if there are 5 actions, all with 3 args, and 12 non-ground facts:
      use algorithm −1000
    - else if all facts ground and 10th/11th domain name letters “PA”:
      use algorithm −1004
    - else if there are 11 actions and action name lengths range from 5 to 28:
      use algorithm 107

Parameter Optimization (2)

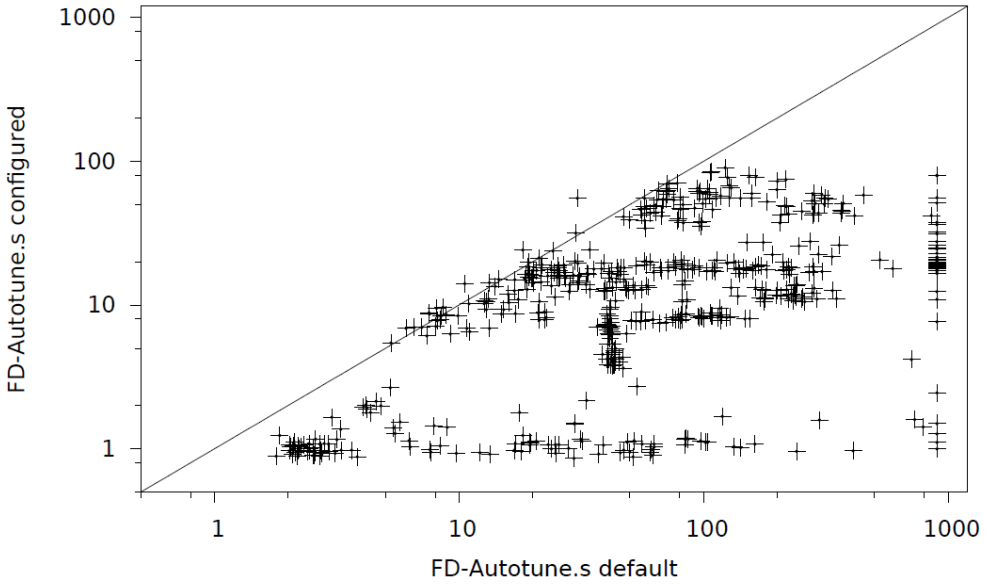
jonkv@ida

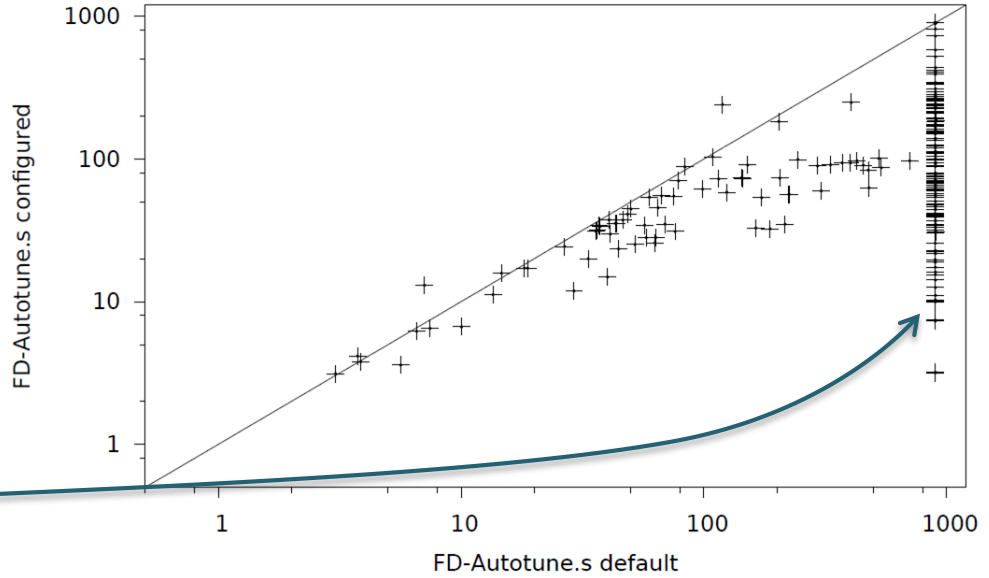
- How about *automatically* learning parameters?
  - One specific form of learning in planning – others exist
  - Experimental application to **Fast Downward**
    - Optimization for speed: 45 params, $2.99 * 10^{13}$ possible configurations
    - Optimization for quality: 77 params, $1.94 * 10^{26}$ possible configurations
  - Example parameters:
    - **Heuristics used**:
      h_{max} = h_0, h_m, h_{add}, h_{FF}, h_{LM} (landmarks), h_{LA} (admissible landmarks), goal count, …
    - Method used to **combine heuristics**: Max, sum, selective max (learns which heuristic to use per state), tie-breaking, Pareto-optimal, alternation
    - **Preferred operators** used or not, for each heuristic
      - Like FF's helpful actions, but used for *prioritization*, not pruning
    - **Search strategy** combinations: Eager best-first, lazy best-first, EHC
    - …
  - Parameter learning framework **ParamILS** used

Parameter Optimization (3): Results

jonkv@ida

- **Under** the diagonal = **faster** than default configuration
  - For 540 small **training instances**:
    - Very good results
    - To be expected – parameters tuned for these specific instances!
  - For 270 larger **test instances**:
    - From the same domains
    - Performance still improves



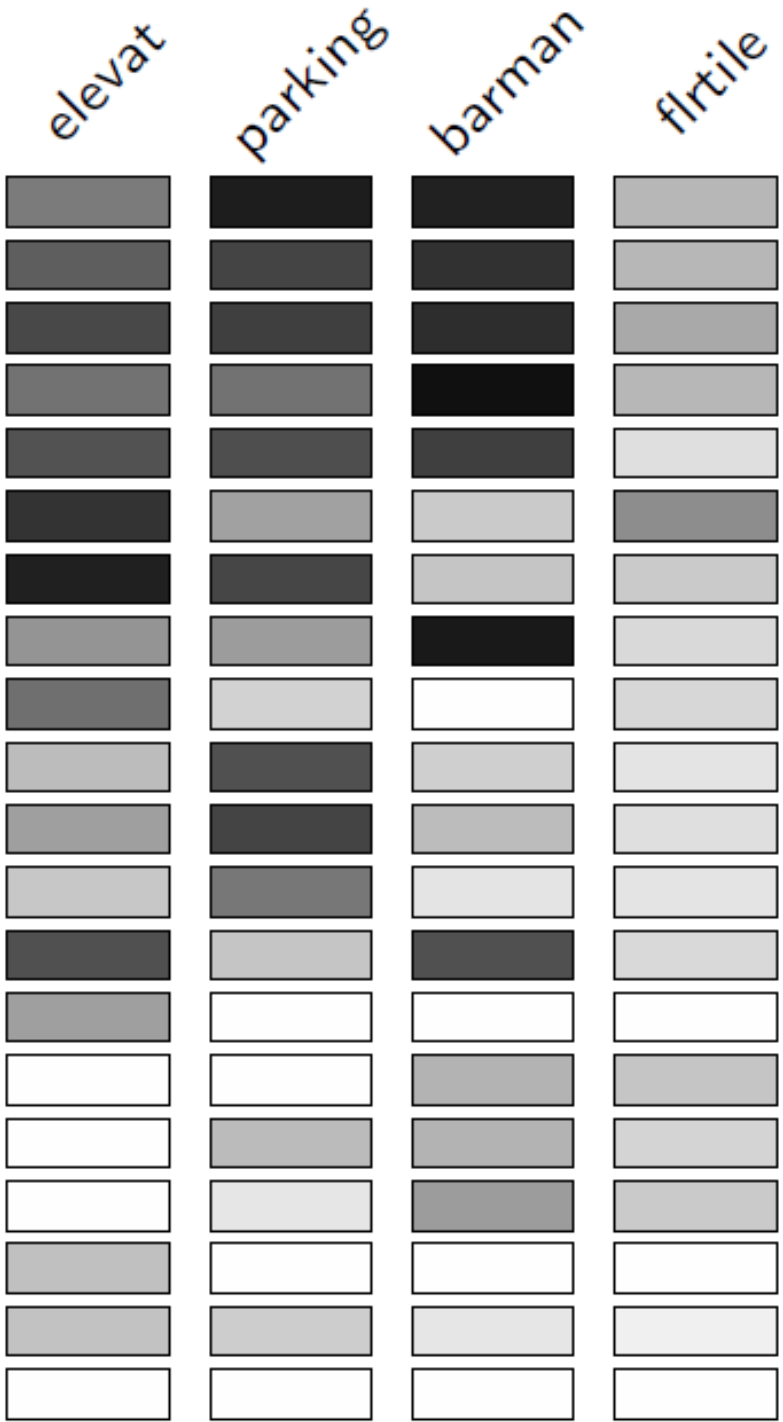


Unsolvable in 900 seconds by the default configuration

Parameter Optimization (4): Results

jonkv@ida

- Complete results



lama-2011
fdss-1
fdss-2
fd-autotune-1
roamer
fd-autotune-2
forkuniform
probe
arvand
lama-2008
lamar
randward
brt
cbp2
daeyahsp
yahsp2
yahsp2-mt
cbp
lprpgp
madagascar-p

Modified LAMA wins!

What's this? Let's see…

Two autotune variations, adapted to *older* domains (a few of which were still used)

Clear winner from 2008, now outrun by others

Portfolio Planners

Portfolio Planning (1)

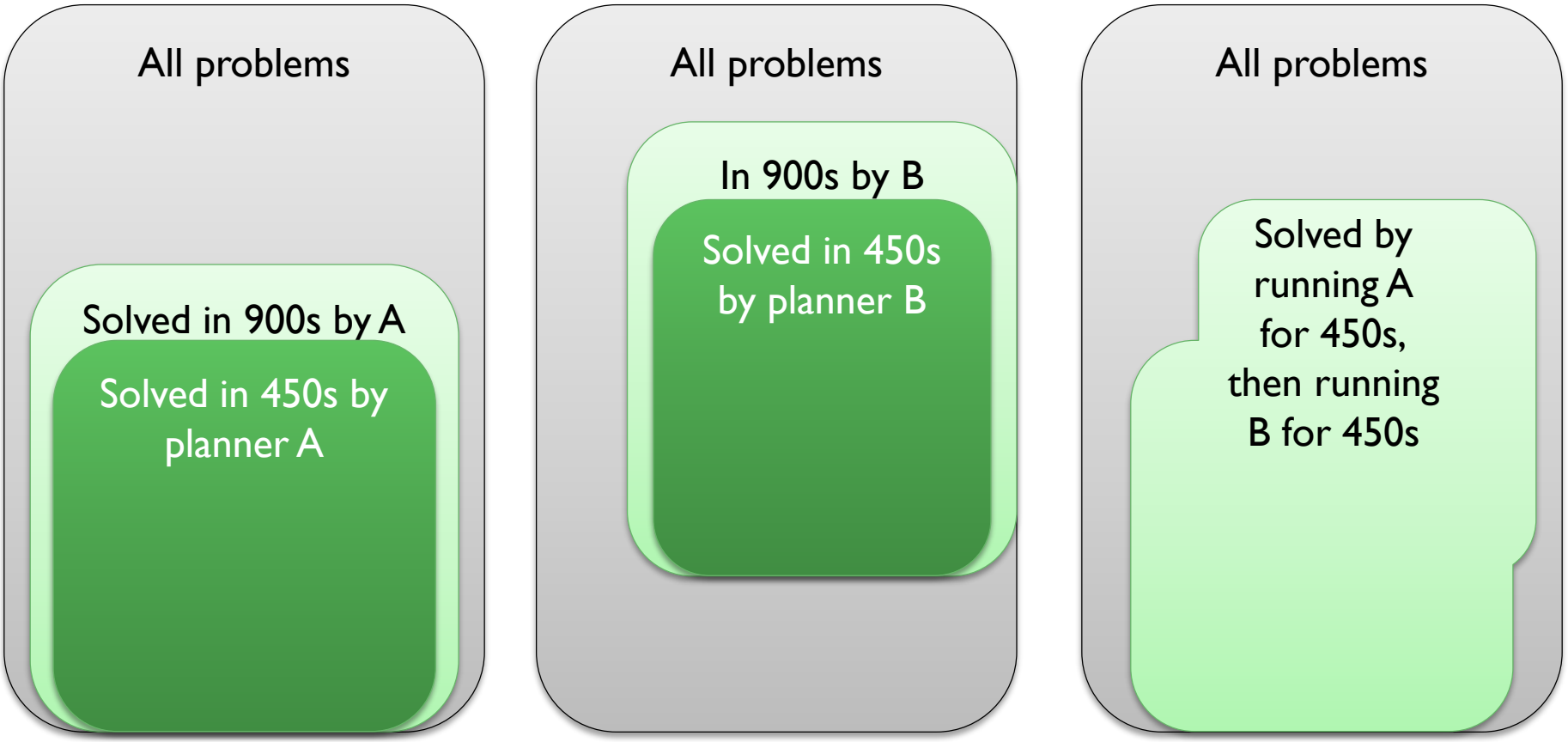
jonkv@ida

- Observation:
  - Different ***planners*** seem good in different ***domains***!

Portfolio Planning (2)

jonkv@ida

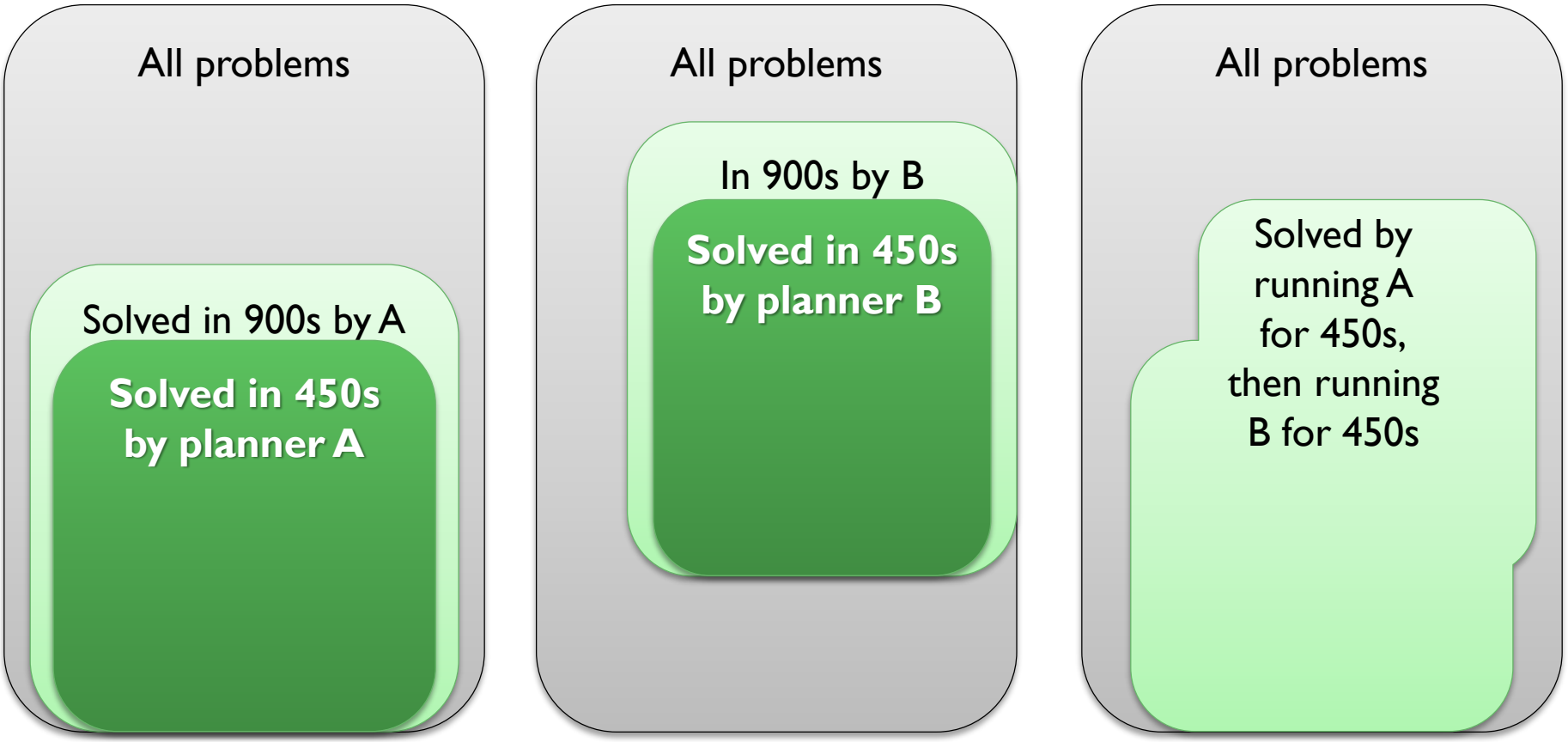
- Further analysis would show:
  - Even if two planners solve equally many problems in one domain, they may solve **different** problems
  - Also, planners often return plans **quickly** or **not at all**



Portfolio Planning (3)

jonkv@ida

- Given a fixed time limit:
  - Can benefit from splitting this across **multiple algorithms**!
  - ➔ **Portfolio** planning



Fast Downward Stone Soup

”We’ll cook a soup using only a stone!
…
But of course it’s tastier if *you* contribute some of this,
and *you* contribute some of that, and…”

FDSS (1)

jonkv@ida

- Fast Downward Stone Soup (2011): **Learning**
  - Given test examples from *earlier* domains (2008)
    - Which *configurations* to use
    - How much *time* to assign to each config

Algorithm	Score	Time	Marginal
BJOLP	605	455	46
RHW landmarks	597	0	—
LM-cut	593	569	26
h^1 landmarks	588	0	—
M&S-bisim 1	447	175	8
$h^{\max}$	427	0	—
M&S-bisim 2	426	432	20
blind	393	0	—
M&S-LFPA 10000	316	0	—
M&S-LFPA 50000	299	0	—
M&S-LFPA 100000	286	0	—
Portfolio	654	1631	
"Holy Grail"	673		

4 of 11 configurations selected for sequential **optimal** planning

FDSS (3)

- For satisficing planning:
  - Far more algorithms and variations to choose from!
  - Lack of time ➔ tested a subset…

Search	Evaluation	Heuristics	Performance	Time	Marg. Contribution
Greedy best-first	Eager	h^{FF}	926.13 / 1021	88	1.82 / 0
Weighted A* ($w = 3$)	Lazy	h^{FF}	921.71 / 1023	340	10.02 / 5
Greedy best-first	Eager	h^{FF}, h^{CG}	919.24 / 1023	76	1.15 / 0
Greedy best-first	Eager	h^{add}, h^{FF}, h^{CG}	909.75 / 1021	0	—
Greedy best-first	Eager	h^{FF}, h^{CG}, h^{cea}	907.52 / 1010	73	1.25 / 0
Greedy best-first	Eager	h^{FF}, h^{cea}	906.92 / 1008	0	—
Greedy best-first	Eager	h^{add}, h^{FF}, h^{CG}, h^{cea}	903.57 / 1012	0	—
Greedy best-first	Eager	h^{add}, h^{FF}	900.52 / 1015	90	1.51 / 1
Greedy best-first	Eager	h^{add}, h^{CG}, h^{cea}	892.08 / 1012	0	—
Greedy best-first	Eager	h^{add}, h^{FF}, h^{cea}	890.96 / 1002	0	—
Greedy best-first	Eager	h^{CG}, h^{cea}	889.93 / 1009	0	—
Greedy best-first	Eager	h^{add}, h^{CG}	888.64 / 1014	0	—
Greedy best-first	Lazy	h^{FF}	880.12 / 1042	171	7.24 / 9
Greedy best-first	Eager	h^{cea}	878.58 / 990	84	3.45 / 2
Greedy best-first	Eager	h^{add}, h^{cea}	877.41 / 999	0	—
Greedy best-first	Lazy	h^{FF}, h^{CG}, h^{cea}	874.64 / 1035	0	—
Weighted A* ($w = 3$)	Eager	h^{FF}	874.18 / 920	87	2.75 / 0
Greedy best-first	Eager	h^{add}	872.74 / 1006	0	—
Greedy best-first	Lazy	h^{FF}, h^{cea}	872.48 / 1037	0	—
Greedy best-first	Lazy	h^{FF}, h^{CG}	871.77 / 1045	49	1.93 / 2
Greedy best-first	Lazy	h^{add}, h^{FF}, h^{CG}, h^{cea}	861.06 / 1032	0	—
Greedy best-first	Lazy	h^{add}, h^{FF}, h^{cea}	860.64 / 1031	0	—
Greedy best-first	Lazy	h^{add}, h^{FF}, h^{CG}	860.04 / 1042	0	—
Greedy best-first	Lazy	h^{add}, h^{FF}	859.72 / 1046	0	—
Weighted A* ($w = 3$)	Lazy	h^{cea}	849.66 / 1001	0	—
Weighted A* ($w = 3$)	Eager	h^{cea}	844.67 / 938	0	—
Greedy best-first	Lazy	h^{CG}, h^{cea}	841.78 / 1026	27	1.25 / 0
Greedy best-first	Lazy	h^{add}, h^{cea}	839.60 / 1020	0	—
Greedy best-first	Lazy	h^{add}, h^{CG}, h^{cea}	835.33 / 1019	0	—
Greedy best-first	Lazy	h^{add}, h^{CG}	831.28 / 1030	0	—
Weighted A* ($w = 3$)	Lazy	h^{add}	830.39 / 1006	50	0.90 / 0
Weighted A* ($w = 3$)	Eager	h^{add}	828.76 / 936	166	3.35 / 3
Greedy best-first	Lazy	h^{cea}	827.57 / 1014	56	2.04 / 2
Weighted A* ($w = 3$)	Eager	h^{CG}	822.46 / 906	89	2.30 / 1
Greedy best-first	Lazy	h^{add}	808.80 / 1019	0	—
Greedy best-first	Eager	h^{CG}	802.47 / 920	0	—
Weighted A* ($w = 3$)	Lazy	h^{CG}	782.14 / 908	73	2.57 / 1
Greedy best-first	Lazy	h^{CG}	755.43 / 924	0	—
Portfolio			1057.57 / 1071	1519	
"Holy Grail"			1078.00 / 1078		

Search	Evaluation	Heuristics	Performance	Time	Marg. Contribution
Greedy best-first	Eager	h^{FF}	926.13 / 1021	88	1.82 / 0
Weighted A* ($w = 3$)	Lazy	h^{FF}	921.71 / 1023	340	10.02 / 5
Greedy best-first	Eager	h^{FF}, h^{CG}	919.24 / 1023	76	1.15 / 0
Greedy best-first	Eager	h^{add}, h^{FF}, h^{CG}	909.75 / 1021	0	—
Greedy best-first	Eager	h^{FF}, h^{CG}, h^{cea}	907.52 / 1010	73	1.25 / 0
Greedy best-first	Eager	h^{FF}, h^{cea}	906.92 / 1008	0	—
Greedy best-first	Eager	h^{add}, h^{FF}, h^{CG}, h^{cea}	903.57 / 1012	0	—
Greedy best-first	Eager	h^{add}, h^{FF}	900.52 / 1015	90	1.51 / 1
Greedy best-first	Eager	h^{add}, h^{CG}, h^{cea}	892.08 / 1012	0	—
Greedy best-first	Eager	h^{add}, h^{FF}, h^{cea}	890.96 / 1002	0	—
Greedy best-first	Eager	h^{CG}, h^{cea}	889.93 / 1009	0	—
Greedy best-first	Eager	h^{add}, h^{CG}	888.64 / 1014	0	—
Greedy best-first	Lazy	h^{FF}	880.12 / 1042	171	7.24 / 9
Greedy best-first	Eager	h^{cea}	878.58 / 990	84	3.45 / 2
Greedy best-first	Eager	h^{add}, h^{cea}	877.41 / 999	0	—
Greedy best-first	Lazy	h^{FF}, h^{CG}, h^{cea}	874.64 / 1035	0	—
Weighted A* ($w = 3$)	Eager	h^{FF}	874.18 / 920	87	2.75 / 0
Greedy best-first	Eager	h^{add}	872.74 / 1006	0	—
Greedy best-first	Lazy	h^{FF}, h^{cea}	872.48 / 1037	0	—
Greedy best-first	Lazy	h^{FF}, h^{CG}	871.77 / 1045	49	1.93 / 2
Greedy best-first	Lazy	h^{add}, h^{FF}, h^{CG}, h^{cea}	861.06 / 1032	0	—
Greedy best-first	Lazy	h^{add}, h^{FF}, h^{cea}	860.64 / 1031	0	—
Greedy best-first	Lazy	h^{add}, h^{FF}, h^{CG}	860.04 / 1042	0	—
Greedy best-first	Lazy	h^{add}, h^{FF}	859.72 / 1046	0	—
Weighted A* ($w = 3$)	Lazy	h^{cea}	849.66 / 1001	0	—
Weighted A* ($w = 3$)	Eager	h^{cea}	844.67 / 938	0	—
Greedy best-first	Lazy	h^{CG}, h^{cea}	841.78 / 1026	27	1.25 / 0

Greedy best-first	Lazy	h^{add}, h^{cea}	839.60 / 1020	0	—
Greedy best-first	Lazy	h^{add}, h^{CG}, h^{cea}	835.33 / 1019	0	—
Greedy best-first	Lazy	h^{add}, h^{CG}	831.28 / 1030	0	—
Weighted A* ($w = 3$)	Lazy	h^{add}	830.39 / 1006	50	0.90 / 0
Weighted A* ($w = 3$)	Eager	h^{add}	828.76 / 936	166	3.35 / 3
Greedy best-first	Lazy	h^{cea}	827.57 / 1014	56	2.04 / 2
Weighted A* ($w = 3$)	Eager	h^{CG}	822.46 / 906	89	2.30 / 1
Greedy best-first	Lazy	h^{add}	808.80 / 1019	0	—
Greedy best-first	Eager	h^{CG}	802.47 / 920	0	—
Weighted A* ($w = 3$)	Lazy	h^{CG}	782.14 / 908	73	2.57 / 1
Greedy best-first	Lazy	h^{CG}	755.43 / 924	0	—
Portfolio			1057.57 / 1071	1519	
"Holy Grail"			1078.00 / 1078		

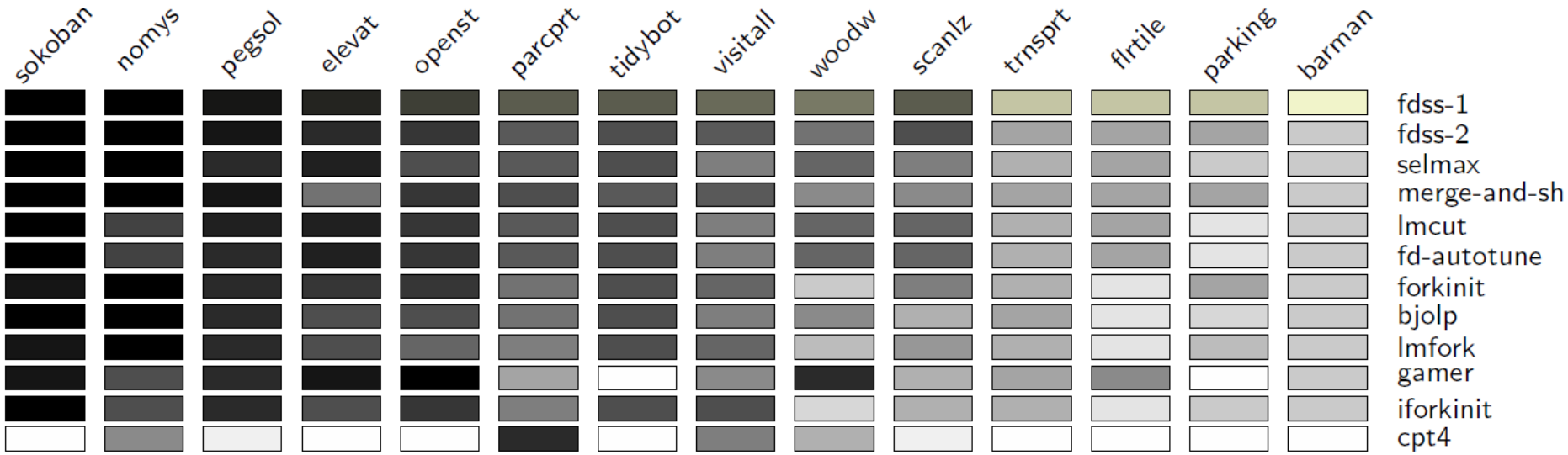
Results: 2011

http://www.plg.inf.uc3m.es/ipc2011-deterministic/

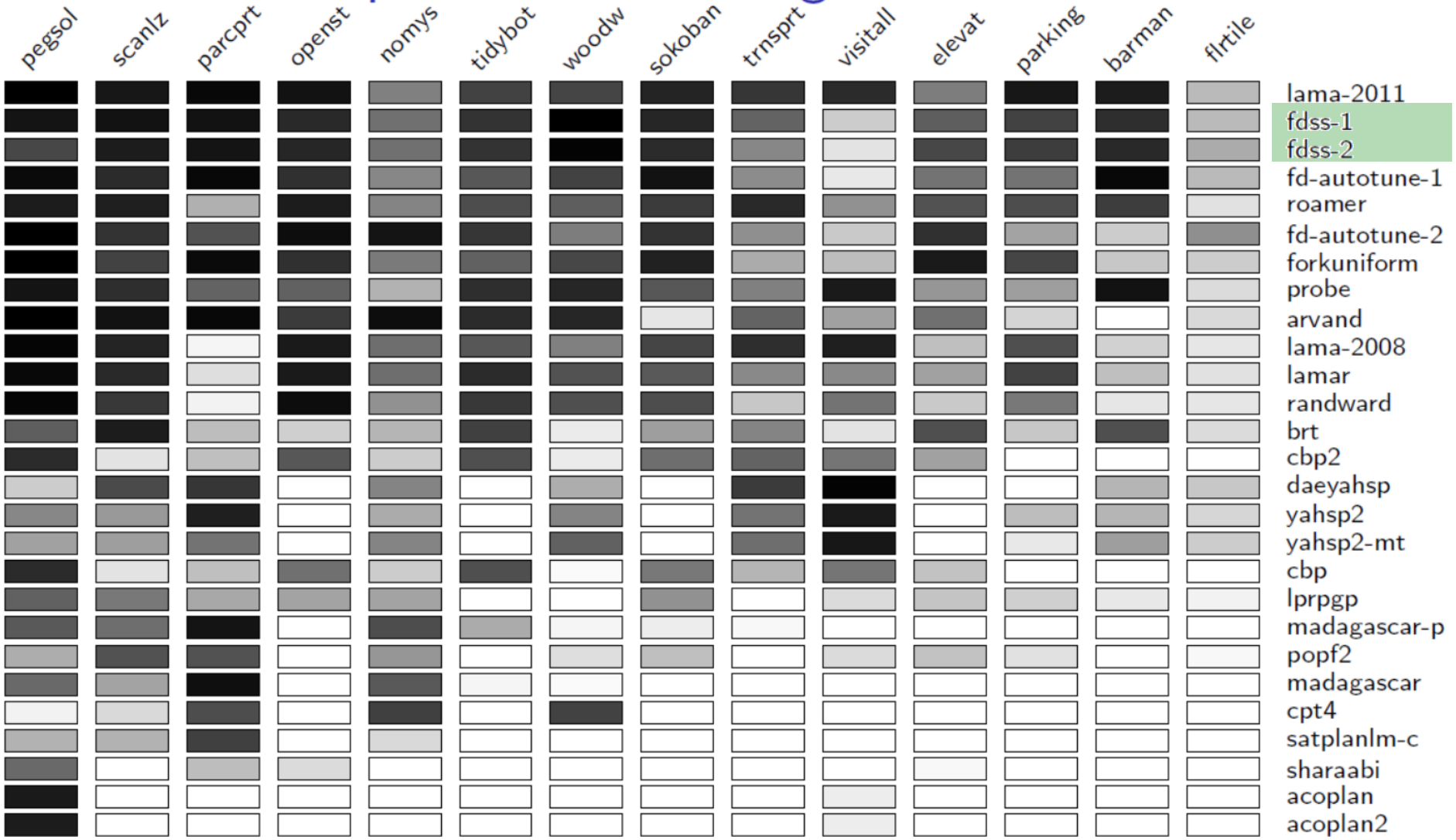
IPC 2011: Optimization

jonkv@ida

- Results from IPC-2011
  - Using new, previously unknown domains



Sequential Satisficing track: Results



2014

What you always wanted to know about the deterministic part of the International Planning Competition (IPC) 2014 (but were too afraid to ask)

MAURO VALLATI[1], LUKÁŠ CHRPA[2,3] and THOMAS L. MCCLUSKEY[4]

Table 3 International Planning Competition (IPC) score, number of problems solved, success rate, Borda score and average Borda score for the *sequential satisficing* track participants

Planner	Score	# Solved	% Solved	Borda	Average Borda
IBaCoP2	**166.2**	**198**	**70.7**	205	14.6
IBaCoP	162.7	196	70.0	**206**	**14.7**
Mercury	153.0	172	61.4	189	13.5
MIPlan	150.0	168	60.0	192	13.7
Jasper	144.9	173	61.8	201	14.4
FD-Uniform	143.3	172	61.4	196	14.0
FD-Cedalion	137.3	160	57.1	189	13.5
ArvandHerd	137.1	158	56.4	197	14.1
FDSS-2014	127.9	151	53.9	184	13.1
DPMPlan	125.5	147	52.5	158	11.3
USE	107.1	163	58.2	140	10.0
NuCeLaR	101.4	122	43.6	162	11.6
RPT	98.3	127	45.4	144	10.3
BFS(f)	96.1	104	37.1	146	10.4
BiFD	87.0	112	40.0	130	9.3
DAE-YAHSP	64.2	100	35.7	107	7.6
Freelunch	61.2	110	39.3	111	7.9
YAHSP3-MT	58.5	118	42.1	104	7.4
YAHSP3	48.1	92	32.9	95	6.8
Planets	25.0	26	9.3	72	5.1

Bold indicates best result, according to the corresponding metrics. Solvers are ordered following IPC score.

Table 3 International Planning Competition (IPC) score, number of problems solved, success rate, Borda score and average Borda score for the *sequential satisficing* track participants

Planner	Score	# Solved	% Solved	Borda	Average Borda
IBaCoP2	**166.2**	**198**			
IBaCoP	162.7	196			
Mercury	153.0	172			
MIPlan	150.0	168			
Jasper	144.9	173			
FD-Uniform	143.3	172			
FD-Cedalion	137.3	160			
ArvandHerd	137.1	158			
FDSS-2014	127.9	151			
DPMPlan	125.5	147			
USE	107.1	163			
NuCeLaR	101.4	122			
RPT	98.3	127			
BFS(f)	96.1	104			
BiFD	87.0	112			
DAE-YAHSP	64.2	100			
Freelunch	61.2	110			
YAHSP3-MT	58.5	118			
YAHSP3	48.1	92			
Planets	25.0	26			

No LAMA here?

Completely outdated?

Part of many portfolios, including the two winners…

Bold indicates best result, according to the corresponding metrics. Solvers are ordered following IPC score.

IBaCoP 2014

jonkv@ida

- IBaCoP: Instance-Based Configured Portfolios
  - #1 and #2 in the Sequential Satisficing Track 2014
  - based on:
    - ARVAND (Nakhost, Valenzano, and Xie 2011)
    - **FD-AUTOTUNE 1 & 2** (Fawcett et al. 2011)
    - **FD STONE SOUP (FDSS) 1 & 2** (Helmert et al. 2011)
    - **LAMA 2008 & 2011** (Richter, Westphal, and Helmert2011)
    - PROBE (Lipovetzky and Geffner 2011)
    - MADAGASCAR (Rintanen 2011)
    - RANDWARD (Olsen and Bryce 2011)
    - YAHSP2-MT (Vidal 2011)
    - LPG-TD (Gerevini et al. 2004)

2018

https://ipc2018-classical.bitbucket.io/

Results: Satisficing, IPC 2018

jonkv@ida

LAPKT-POLYNOMIAL-BFWS	13.30	17.84	11.92	8.82	10.65	5.00	9.62	2.33	8.63	2.10	90.20
IBaCoP-2018	2014: #1-2	5.83	**14.66**	0	6.40	8.75	0	7.95	8.80	**10.33**	73.56
IBaCoP2-2018		5.95	11.53	0	7.18	8.00	0	7.91	7.20	7.23	66.84
MERWIN	10.83	0	10.10	0	5.82	**12.0**	0	4.94	11.54	7.71	62.93
mercury2014	2014: #3	0	8.67	0	5.82	11.00	0	3.52	12.23	7.69	61.65
LAPKT-DFS+	10.55	12.80	6.72	1.00	5.00	5.00	7.81	4.21	7.79	0	60.89
fs-sim	11.47	5.95	5.16	0	10.47	4.00	0	7.94	4.70	3.90	53.60
fs-blind	3.00	6.00	5.47	0	12.14	4.00	0	7.06	5.32	7.33	50.32
freelunch-doubly-relaxed	7.99	1.57	0.19	0	0	6.00	0	5.57	0.72	0	22.04
freelunch-madagascar	0	2.32	1.96	0	3.00	8.00	0	1.00	0	0	16.27
Symple-2	1.00	2.00	0	0	2.00	5.00	0	0	0	1.00	11.00
Symple-1	1.00	2.00	0	0	2.00	5.00	0	0	0	1.00	11.00
alien	4.24	0	1.12	0	0	4.00	0	0	0	0	9.36

Results: Satisficing, IPC 2018

jonkv@ida

Score	Agricola	Caldera	Data Network	flashfill	Nurikabe	Organic Synthesis	Settlers	Snake	Spider	Termes	SUM
Fast Downward Stone Soup 2018	Portfolio	13.97	10.73	13.74	18.43	9.75	**16.6**	7.20	10.67	8.68	123.27
Fast Downward Remix		14.92	10.58	12.85	**18.61**	9.00	16.21	7.24	10.78	6.95	120.69
LAPKT-DUAL-BFWS	New strategy!		11.79	**16.66**	14.79	11.75	6.73	9.34	12.12	5.80	119.62
Saarplan	Portfolio	11.76	12.47	13.56	16.64	11.00	9.90	8.36	10.93	7.77	116.87
DecStar	12.42	13.64	9.99	13.74	12.42	9.75	15.68	4.03	12.79	6.75	111.21
Cerberus	10.62	10.92	11.71	9.74	15.50	**12.0**	9.87	5.72	13.94	7.94	107.95
Baseline: LAMA 2011	!!! 9.53	13.90	7.48	13.74	10.59	**12.0**	15.76	3.55	13.42	7.58	107.56
LAPKT-BFWS-Preference	10.99	15.94	8.18	11.74	12.63	7.00	8.43	**15.78**	10.29	5.46	106.44
Cerberus-gl	10.62	10.92	11.71	9.74	15.50	**12.0**	9.11	5.72	**14.16**	6.64	106.13
OLCFF	13.74	11.76	12.19	0.82	17.92	9.00	0	7.62	11.24	7.79	92.09
LAPKT-POLYNOMIAL-BFWS	13.30	17.84	11.92	8.82	10.65	5.00	9.62	2.33	8.63	2.10	90.20

Fast Downward Stone Soup 2018

jonkv@ida

- FDSS update 2018:
  - 144 Fast Downward configurations to choose from
    - Trained using problems from IPC 1998-2014 + other sources
  - Result:
    - Portfolio uses 41 configurations, between 8 and 135 seconds
    - Overall score 1999.93, compared to best component 1650.40 (running on non-training problems