Linköping University

Automated Planning

Miscellaneous Topics…

Jonas Kvarnström
Automated Planning Group
Department of Computer and Information Science
Linköping University

Deferred Evaluation / Lazy Search

Deferred Evaluation

- Standard greedy best-first search:
  - Remove the "best" state from the priority queue
  - Check whether it satisfies the goal
  - Generate all successors
  - Calculate their heuristic values ← Typically takes most of the time
  - Place in priority queue

- Potentially faster: **Deferred Evaluation** (Fast Downward, …)
  - Remove the "best" state from the priority queue
  - Check whether it satisfies the goal
  - Calculate its heuristic value (only one!)
  - Generate all successors
  - Place in priority queue using the **parent's** heuristic value

Takes less time, but less accurate heuristic – "one step behind"
Often faster but lower-quality plans

Dual Queue Techniques

Dual Queues

- FF calculates **helpful actions**
  - Using its planning-graph-based heuristic
  - Then uses these to **prune** the search tree – **only** uses helpful actions
  - Can be very helpful, but is incomplete
    - ➔ May have to restart without helpful actions

- Fast Downward uses **dual queues**
  - One queue for ordinary successors, one for preferred successors
  - Expansion:
    - Pick the best action from queue 1 (preferred); expand it
    - Pick the best action from queue 2 (non-preferred); expand it
    - Repeat
  - Fewer preferred successors ➔ expanded more often, on average
  - Search remains complete

Boosted Dual Queues

- **Boosted** Dual Queues:
  - Used in later versions of Fast Downward and LAMA
  - Whenever progress is made (new best h-value):
    - Expand 1000 preferred successors
  - If progress is made again within these 1000 successors:
    - Add another 1000, accumulating
    - (Progress made after 300 ➔ keep expanding 1700 more)
  - Still complete, but more aggressive

Parameter Optimization and Portfolio Planners

A general technique – not limited to state-space search!

Parameter Optimization (1)

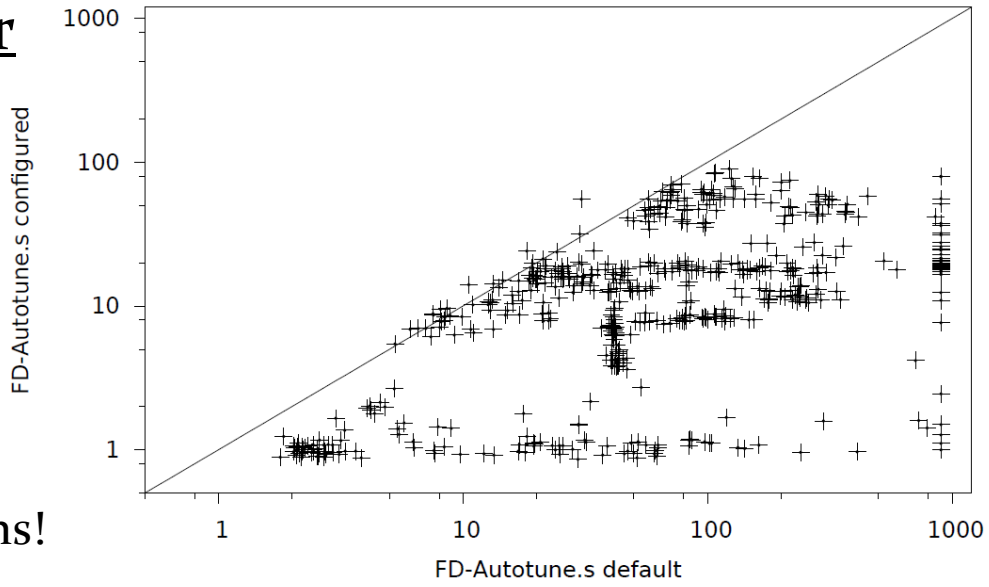
- Some planners have **many parameters** to tweak
  - In early planning competitions, domains were **known in advance**
    - Participants could manually adapt their ”domain-independent” planners…
  - Somewhat **exaggerated citation** from IPC-2008 results:
    - if domain name begins with “PS” and part after first letter is “SR”:
      use algorithm 100
    - else if there are 5 actions, all with 3 args, and 12 non-ground predicates:
      use algorithm –1000
    - else if all predicates 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
  - From 2008, this was no longer allowed
    - Planners were handed in
    - Then the **organizers** ran the planners

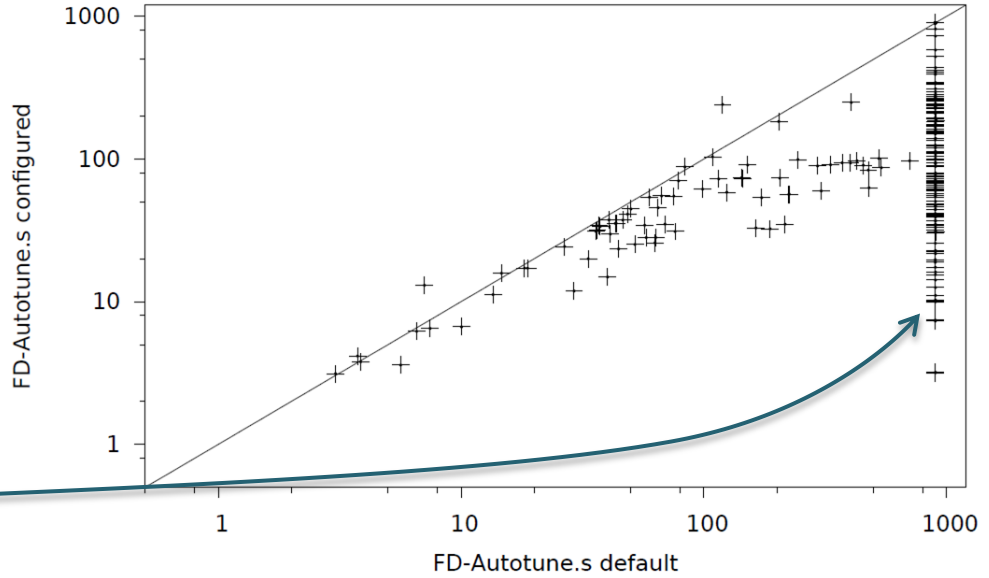
Parameter Optimization (2)

- 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_{cg} (causal graph), h_{cea} (context-enhanced additive), h_{LM} (landmarks), $h_{M\&S}$ (merge-and-shrink), h_{LA} (admissible landmarks), $h_{LM\text{-}cut}$ (admissible landmark-cut), 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

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

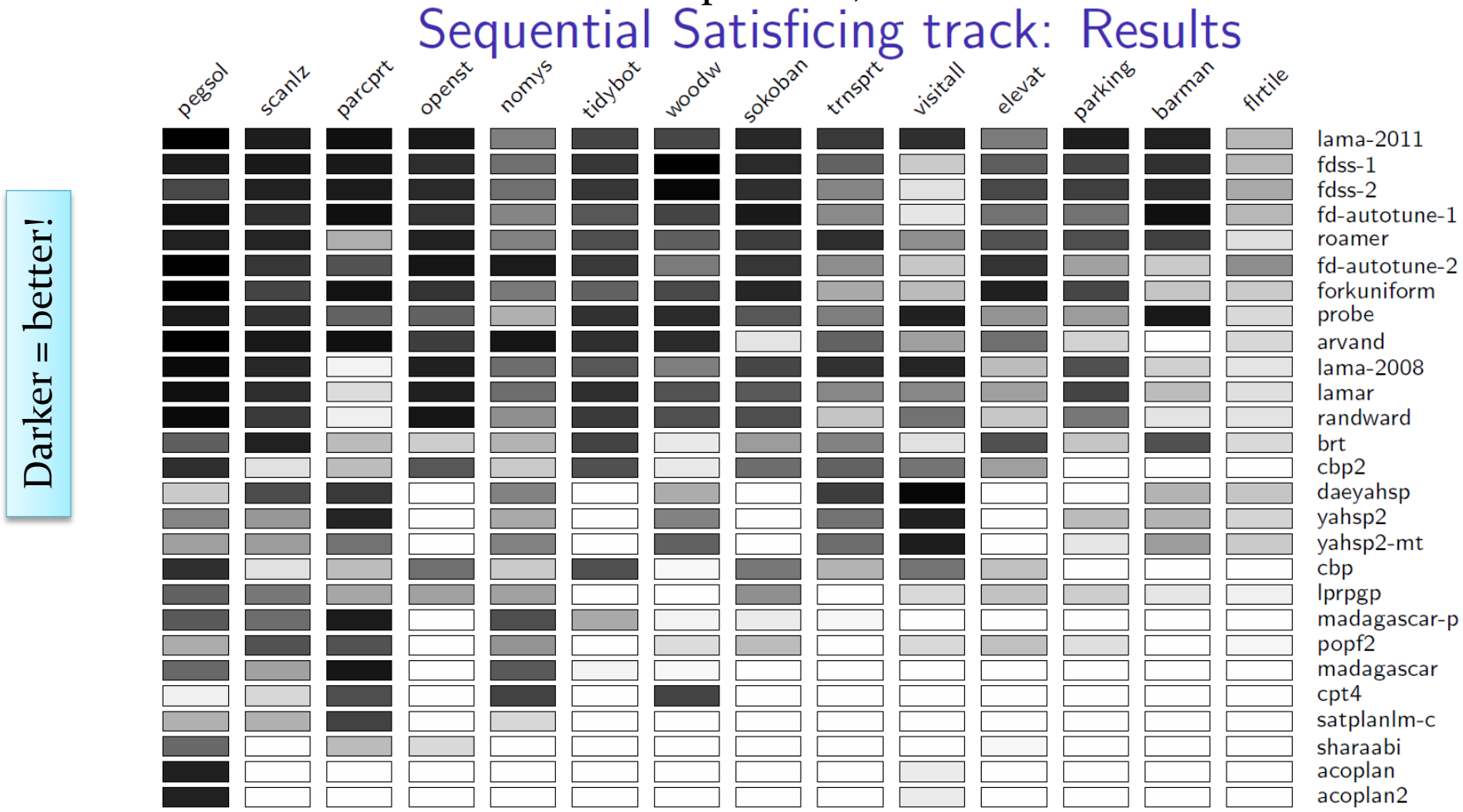




Unsolvable in 900 seconds by the default configuration

Parameter Optimization (4): Results

- Results from the **satisficing** track of IPC-2011
  - Two versions of FD-autotune competed, adapted to *older* domains
  - Some were reused in this competition, most were new

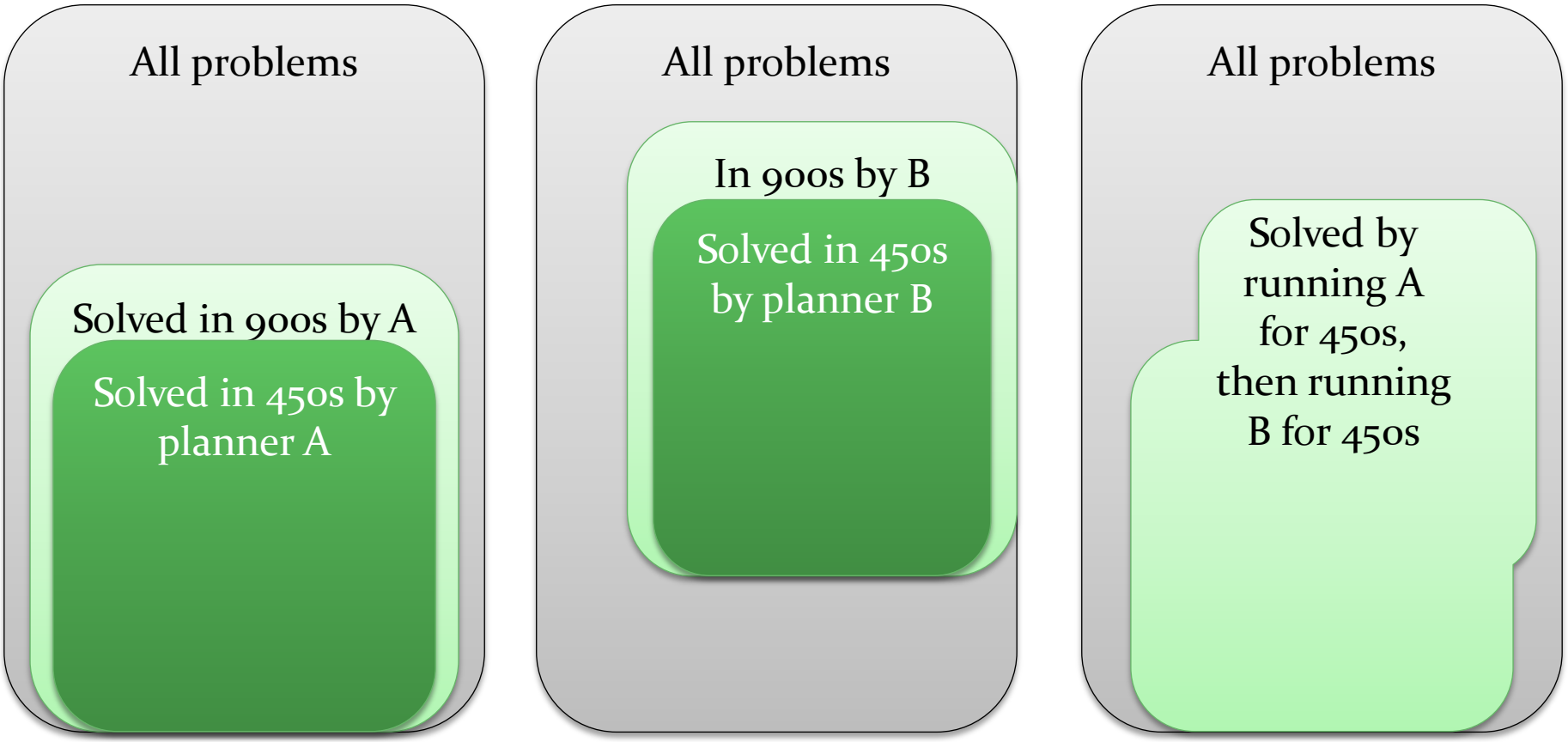


Portfolio Planning (1)

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

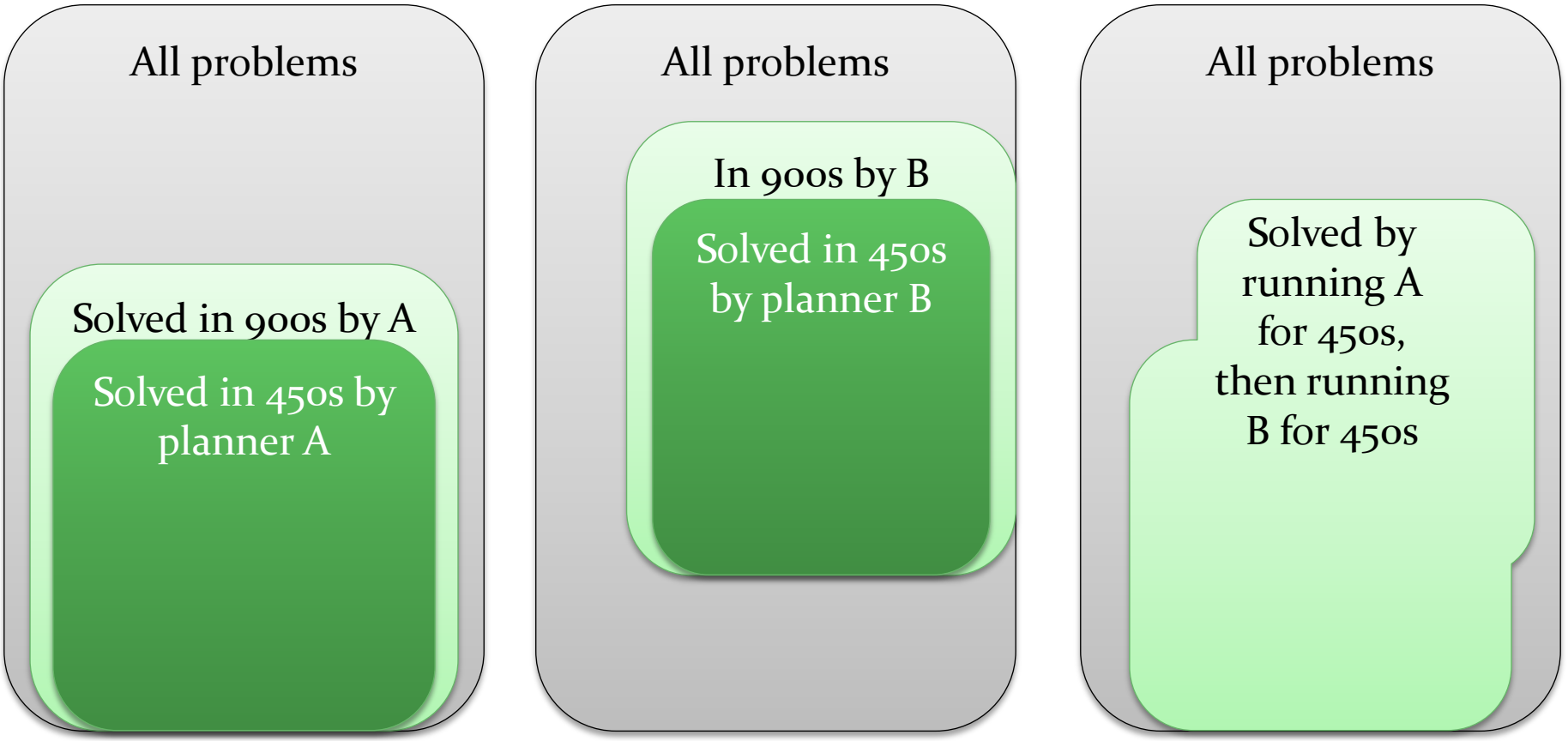
Portfolio Planning (2)

- 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)

- The competition has a fixed time limit
  - Can benefit from splitting this across **multiple algorithms**!
  - ➔ **Portfolio** planning



Portfolio Planning (4)

- Fast Downward Stone Soup: **Learning**
  - Which configurations to use
  - How much time to assign to each one
  - Given test examples from older domains

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		

Configurations learned for sequential optimal planning

Portfolio Planning (5)

- Results from IPC-2011: