



Learning and Local Search

Meinolf Sellmann

joint work with Carlos Ansotegui and
Warren Schudy

Learning and Local Search

- There are two well known connections between Learning and Local Search
 - Local Search, as an optimization technique, can be used for learning purposes
 - Learning can be used to boost local search

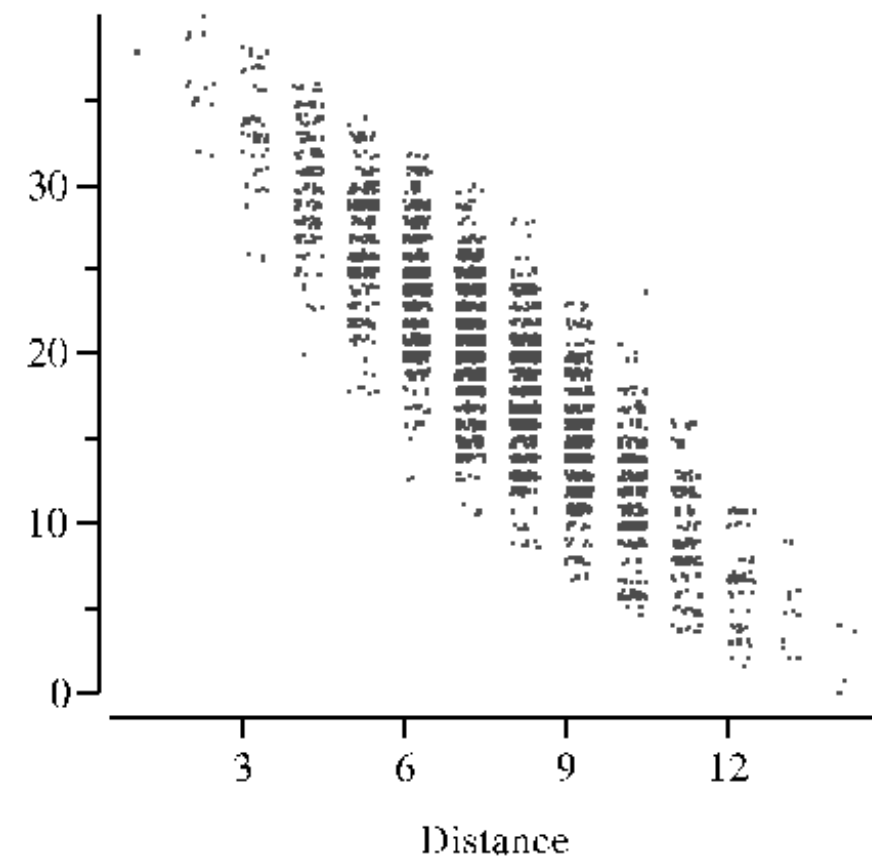
Learning-based Local Search

- Any search-bias must be justified!
- Biased local search performs some form of learning!
- As an incomplete method
 - learning is statistical (non-deterministic)
 - bias must be justified in statistical properties of instances!
- Example: Intensification justified by fitness-distance correlation! [Jones&Forrest '95]

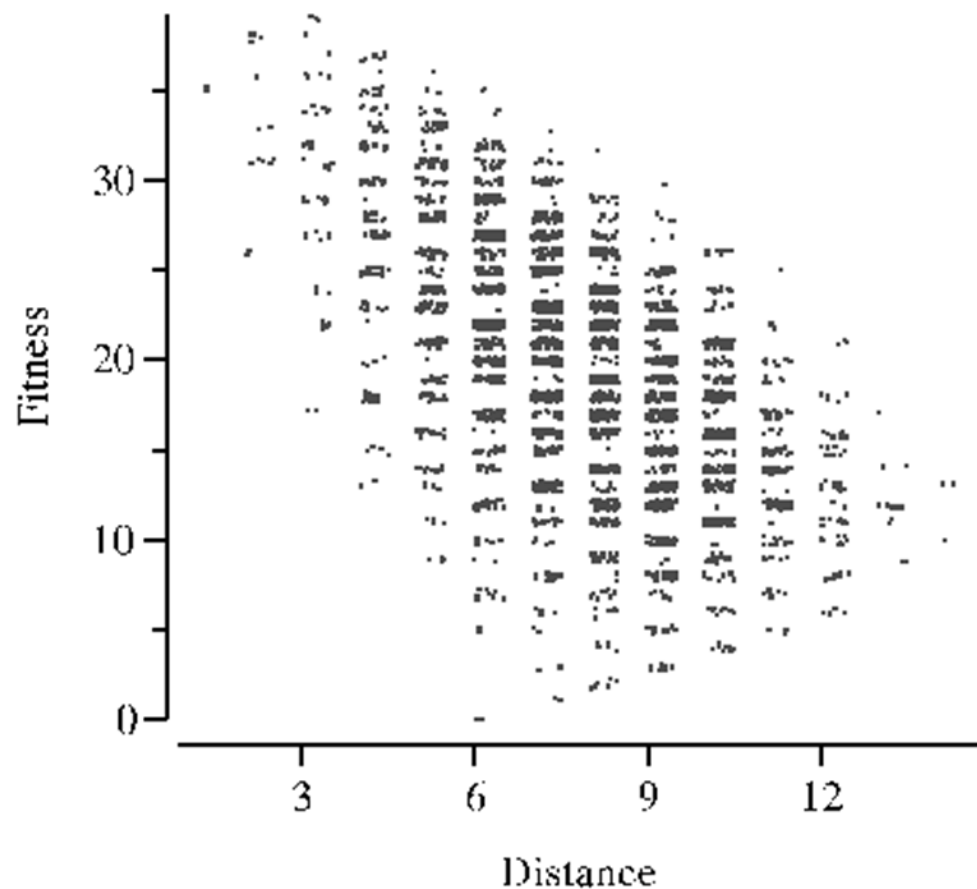
Fitness Distance Correlation

- Take a problem with known global optima.
- Take a sample of points with associated fitnesses $[f_1, \dots, f_n]$ and “distances” (to the nearest optimum) $[d_1, \dots, d_n]$.
- Let \bar{f} , \bar{d} denote the mean and σ_f , σ_d the standard deviation of f and d .
- $c_{fd} := 1/n \sum_i (f_i - \bar{f})(d_i - \bar{d})$
- $r := c_{fd} / \sigma_f \sigma_d$

Fitness Distance Correlation

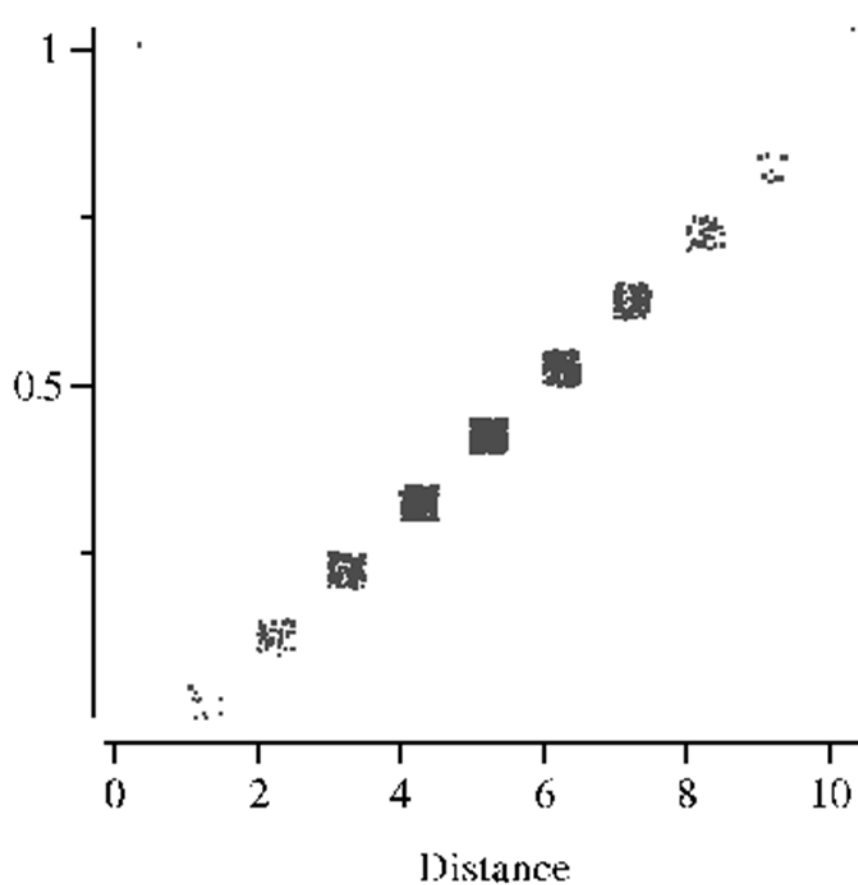


Binary-Code [r=-.86]

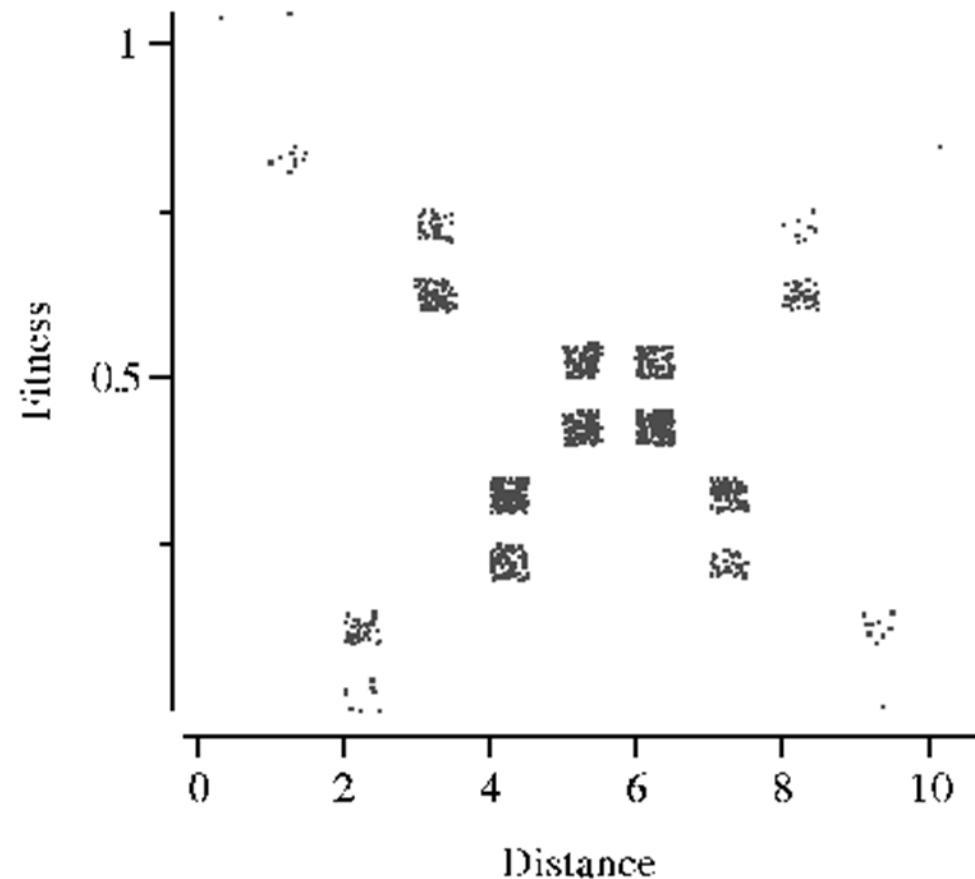


Gray-Code [r=-.57]

Fitness Distance Correlation



Deceptive [$r=0.98$]



Fully-Easy [$r=0$]

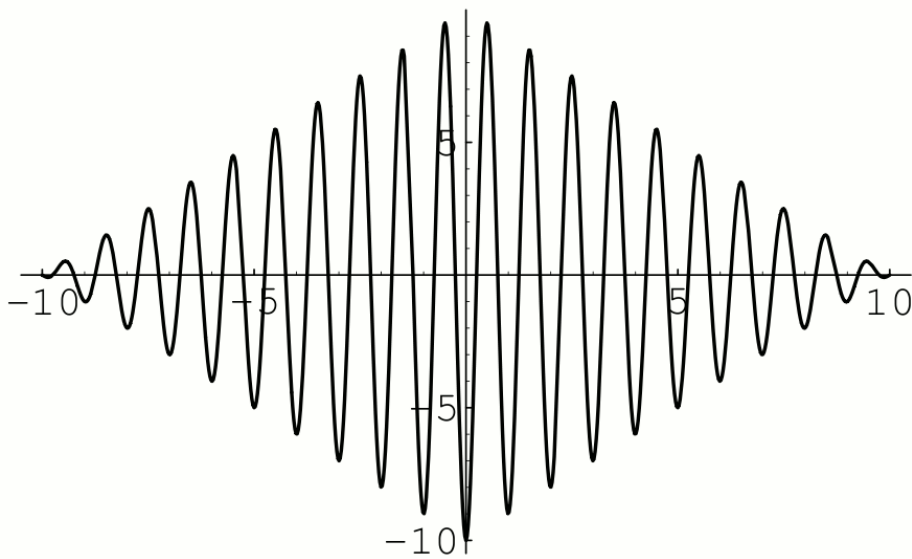
Reactive Tabu-Search

- Problem: Local search may get stuck
 - In local optima
 - In cycles
 - In large sub-regions of the search space
- Idea: Adaptively change the length of the tabu list [Battiti&Tecchioli '94]
 - Store previously visited points
 - Quickly increase length of the tabu list when cycle or many repeated solutions
 - Otherwise slowly decrease list length
 - Randomly escape when many repeated solutions (path length based on cycle length estimate)
- Reported to find better solutions, yet more time-consuming on smaller instances.

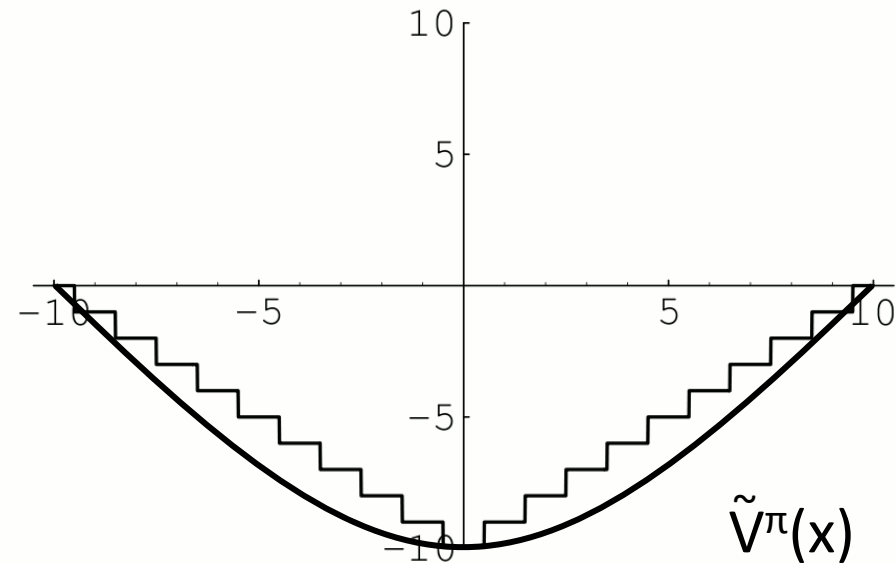
Learning Evaluation Functions

- An adaptive way to learn good evaluation functions online by [Boyan&Moore'00]
- For each potential start point x , learn its **promise** $V^\pi(x)$, the (expected) solution quality of a LS started in x .
- Learning is done by fitting a polynomial through the starting points tried earlier.

Learning Evaluation Functions

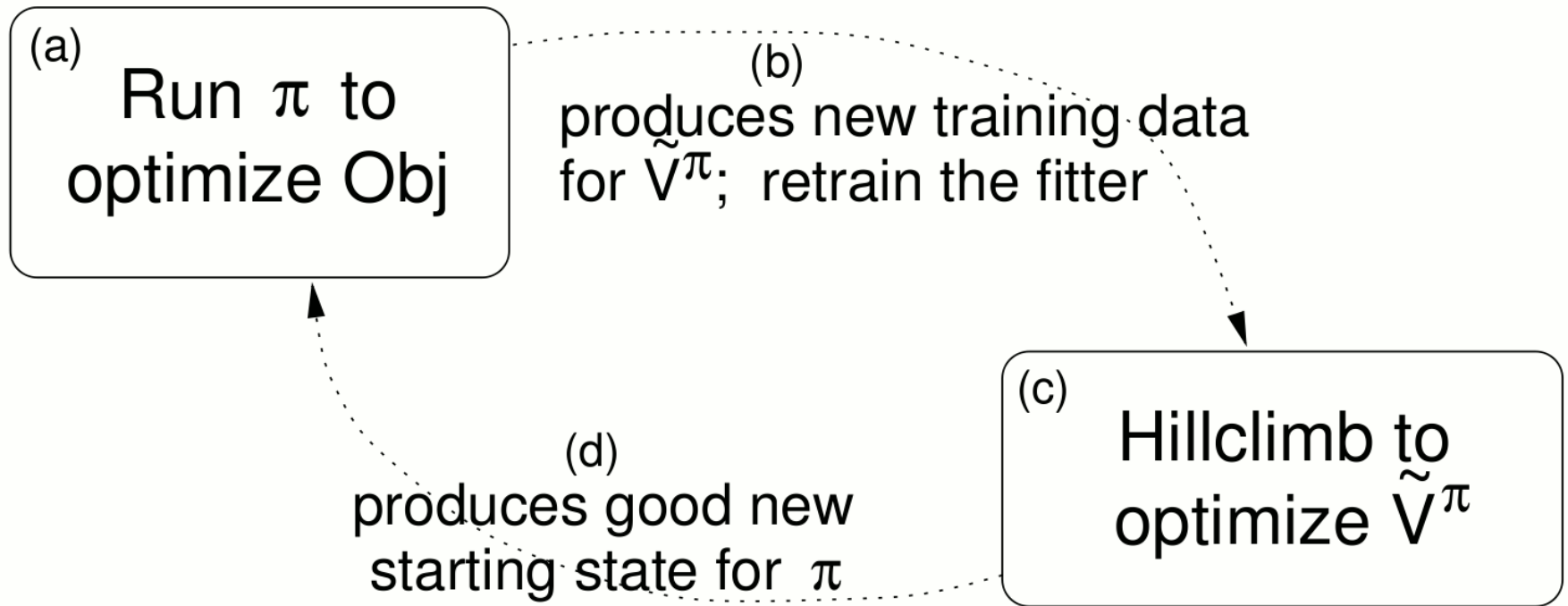


$$(|x| - 10) \cos(2\pi x)$$



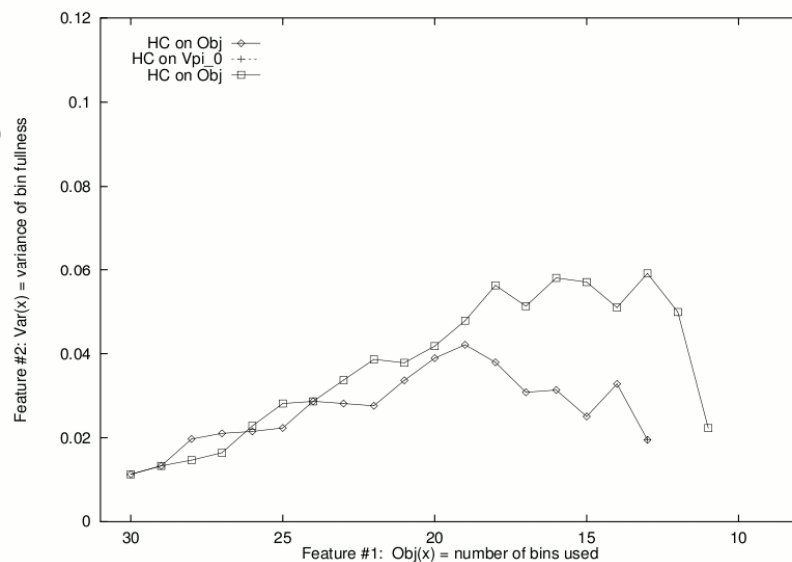
$$V^\pi(x)$$

Learning Evaluation Functions

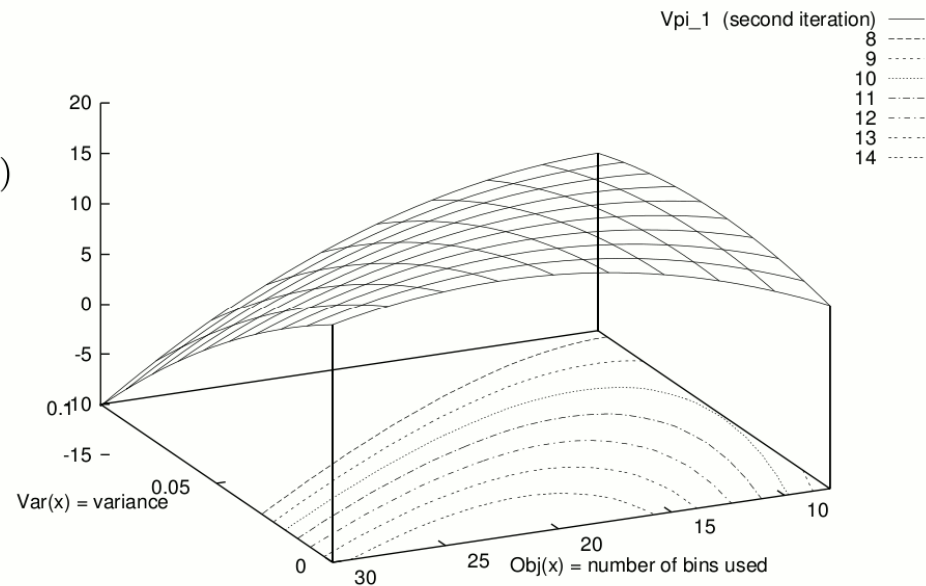


Learning Evaluation Functions

(a)



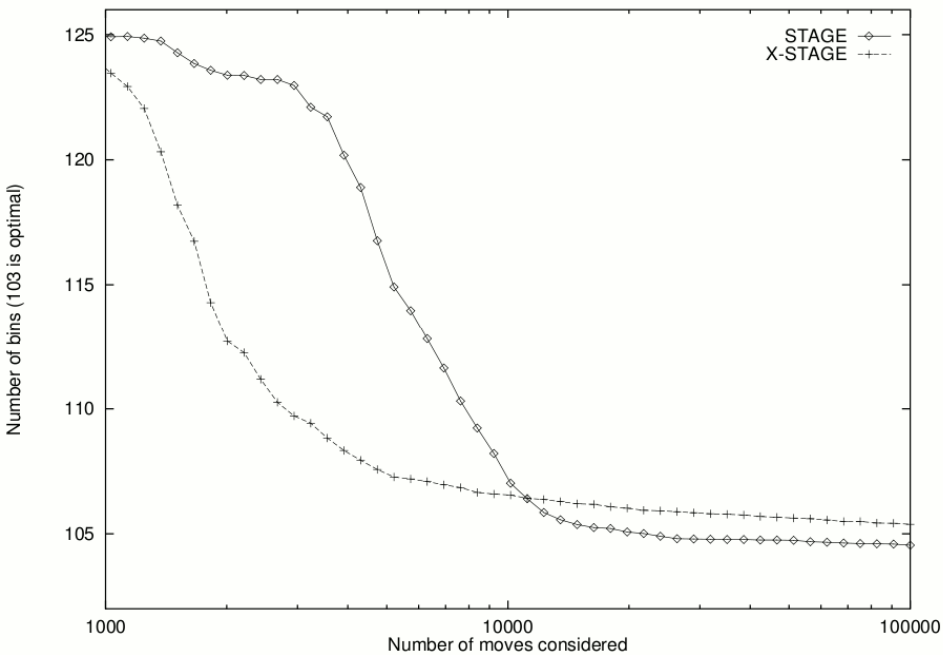
(b)



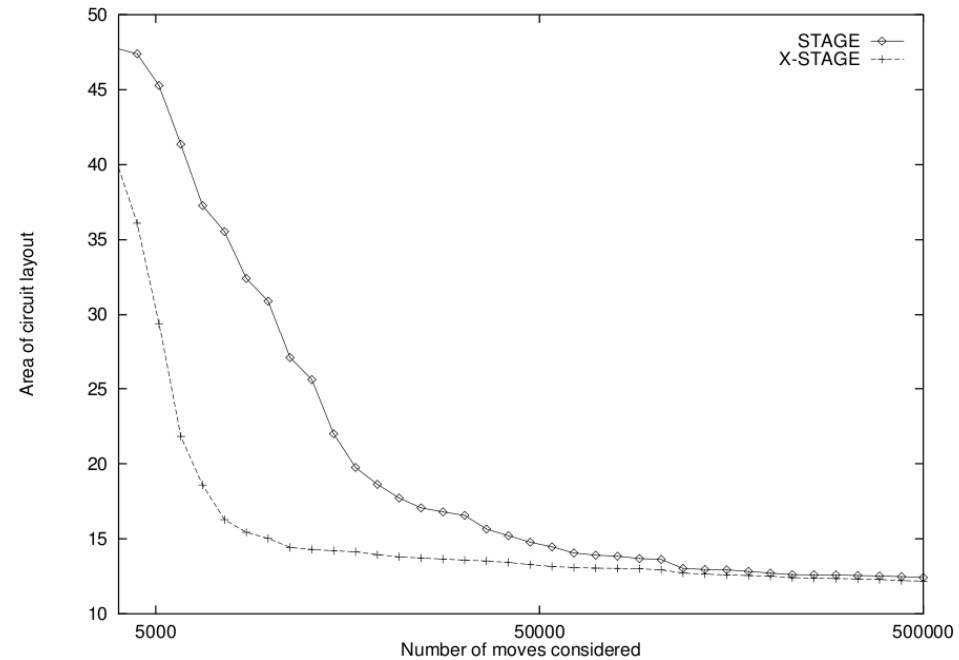
Learning Evaluation Functions

- X-Stage:
 - Learn promise functions from “characteristic” instances
 - To avoid problems due to instance size differences, learn over features of starting points
 - To avoid problems in the output range, have the learned functions vote on the acceptance of a new state

Learning Evaluation Functions



Bin Packing



Channel Routing

Dynamic Local Search and Clause Weighting

- Use global statistics to “fill the holes” which cause local minima.
- Scaling and Probabilistic Smoothing
[Hutter,Tompkins,Hoos’02]
 - Best improvement LS
 - At local minimum
 - Random move with some probability OR
 - Scale unsatisfied clause weights, smooth all weights with some probability
- [Tompkins&Hoos’04] show that new weights make LS not easier, thus questioning any “learning.”
Improvements by DLS are attributed to efficient diversification.

Learning and Local Search

- There are two well known connections between Learning and Local Search
 - Local Search, as an optimization technique, can be used for learning purposes
 - Learning can be used to boost local search

Can Local Search be also viewed and exploited as a “learning” algorithm for systematic search?

Systematic Search

- Systematic Search:
 - How to split the search space?
 - Where to continue searching?
- Special case: Backtrack Search
 - Branching Variable?
 - Order of Branching Values?

Inference and Search

- Note how we use different efficient inference techniques in different areas:
 - Optimization → Relaxations
 - Constraint Programming → Domain Filtering
 - SAT → Unit Propagation and Clause Learning
- In all cases we complement inference with search. While there are strategies for organizing the search (like min-domain or min-integrality), truly robust heuristics are not known.
- Is there an alternative to basing branching decisions on “intuitive” heuristics?

Restarts

- Instead of making intuitively reasonable decisions when organizing the search, better make random decisions.
- Since there is a substantial chance for a very long run, but also a good chance for a shorter run, do the following:
 - Start your search with a limit on the number of fails (backtracks) that are allowed.
 - If it takes longer, increase the fail limit and simply start over, hoping that next time the organization of the search will be more desirable.
 - No-goods learned in previous runs can be saved for the next restart!

Two Schools

- The Believers (Thesis)
 - Search Heuristics
 - Learn from search history
- The Fatalists (Anti-Thesis)
 - Take chances
 - Be aware that bad things can happen
 - Start over when unsuccessful

Related Work

- [Kautz et al AAAI 2002]: Dynamic Restart Policies
- [Balas/Carrera OR 1996]: Randomized Greedy Heuristics
- [Fischetti/Lodi Math Programming 2003]: Local Branching
- [Beck JAIR 2007]: Multi-Point Constructive Search
- [Prestwich 2000]: Stochastic Local Search
- [Epstein et al CP 2002]: Adaptive Search Engine
- [Nudelman et al SAT 2004]: SATzilla
- [Refalo Informs 2006]: Impact-based branching
- [Zanarini/Pesant CP 2007]: Branching based on Solution Counts
- [Braunstein et al. CoRR 2002]: Survey Propagation

Synthesis

- Branching Variable: Randomized
- Branching Value: Learn better orderings

Restarted Solver

Systematic Solver

Randomized
Variable
Selection

Static
Value
Selection

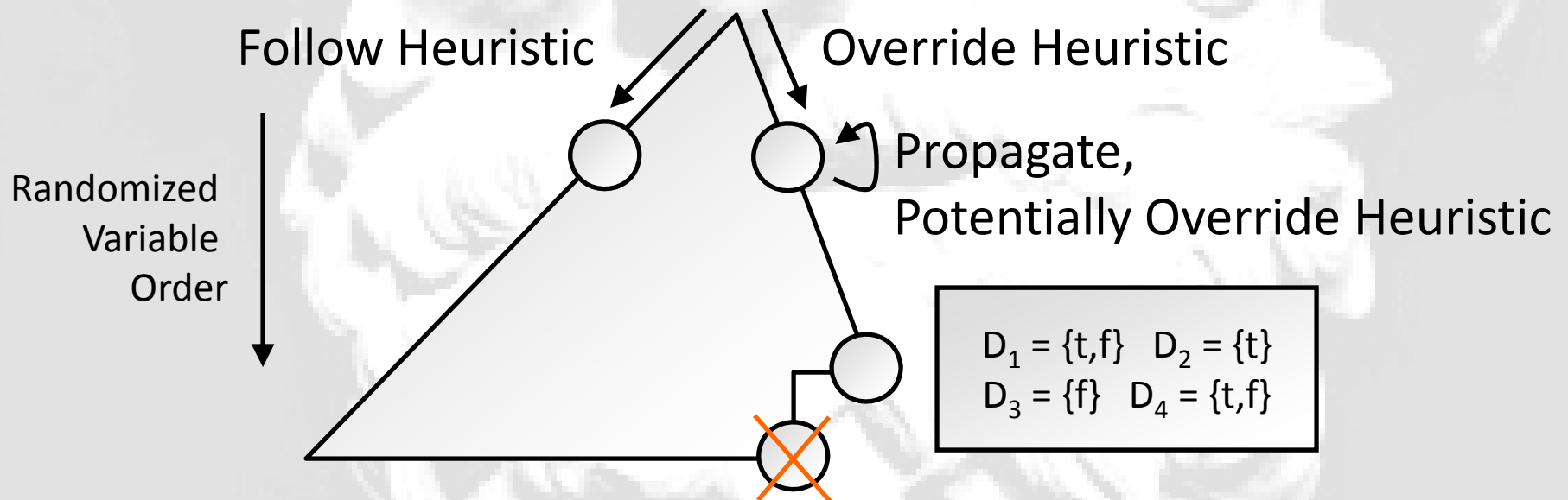


Update Value Heuristic

Update Fail Limit

Update of Value Heuristic

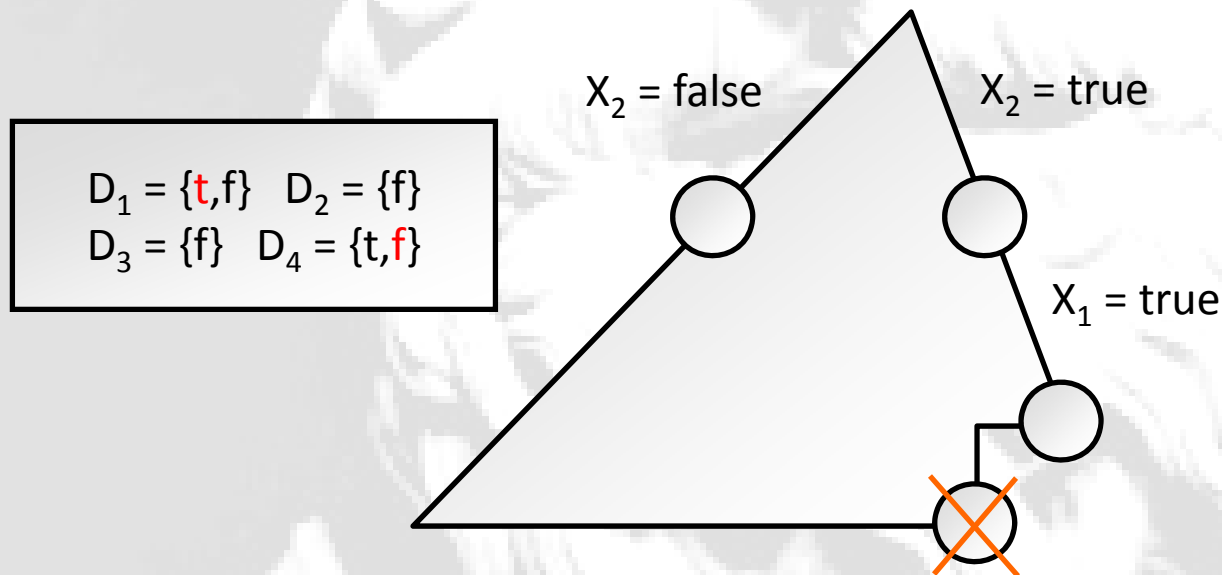
- When the fail limit is reached:



Update of Value Heuristic

- When the fail limit is reached:

Value Heuristic: $X_1 = \text{false}$ $X_2 = \text{false}$ $X_3 = \text{false}$ $X_4 = \text{true}$



$D_1 = \{\text{t}, \text{f}\}$ $D_2 = \{\text{t}, \text{f}\}$
 $D_3 = \{\text{t}, \text{f}\}$ $D_4 = \{\text{t}, \text{f}\}$

$D_1 = \{\text{t}, \text{f}\}$ $D_2 = \{\text{t}\}$
 $D_3 = \{\text{t}, \text{f}\}$ $D_4 = \{\text{f}\}$

$D_1 = \{\text{t}\}$ $D_2 = \{\text{t}\}$
 $D_3 = \{\text{t}, \text{f}\}$ $D_4 = \{\text{f}\}$

Value Heuristic: $X_1 = \text{true}$ $X_2 = \text{true}$ $X_3 = \text{false}$ $X_4 = \text{false}$

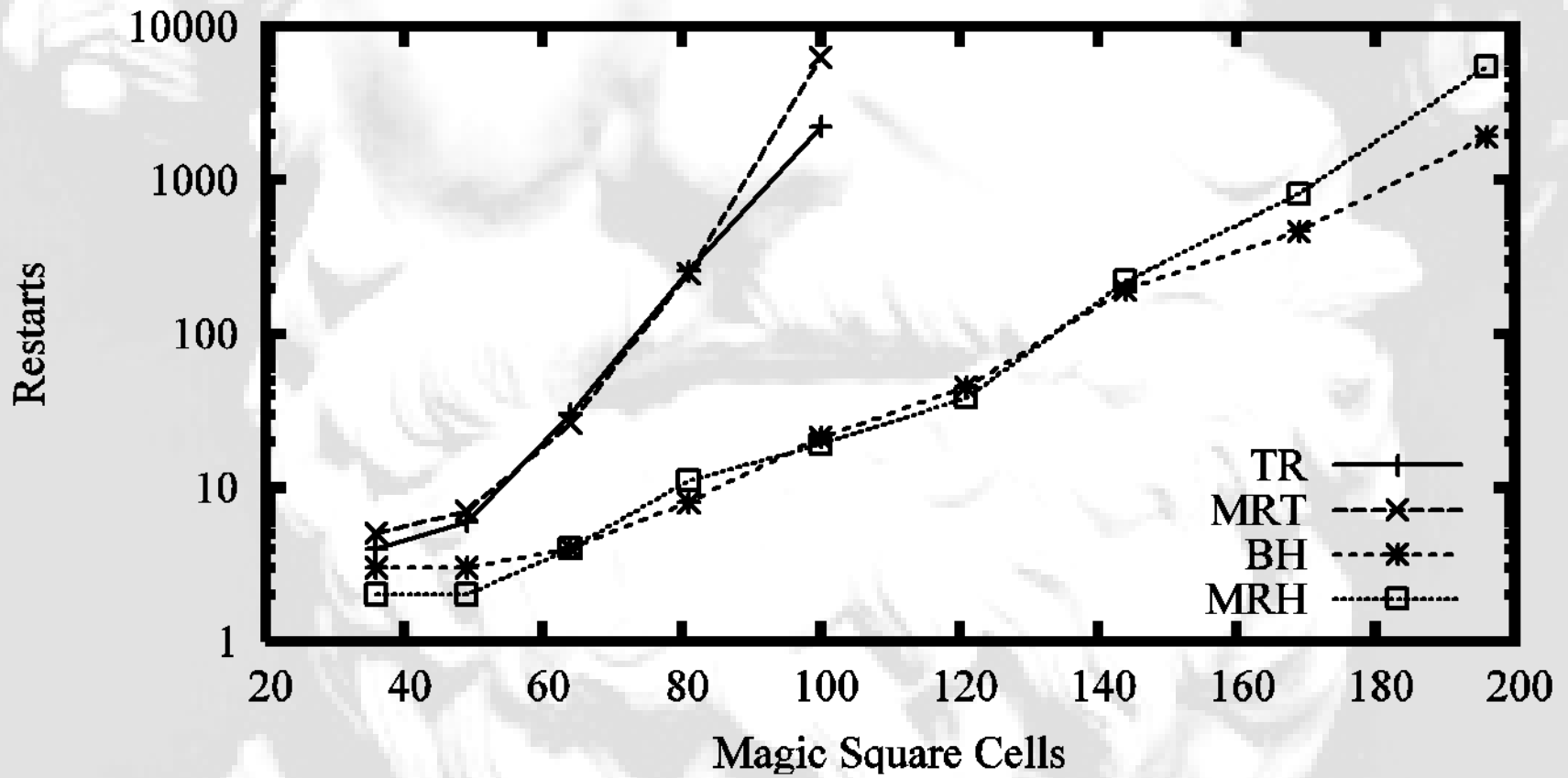
Algorithm

- BH: bool BasicHybrid (void)
 - InitFailLimit(failLimit), InitRandom (heuristic)
 - while (true) do
 - status := TreeSearch(failLimit,heuristic);
 - if (status != inconclusive) then return (status == solved);
 - UpdateHeuristic (heuristic), UpdateFailLimit (failLimit);

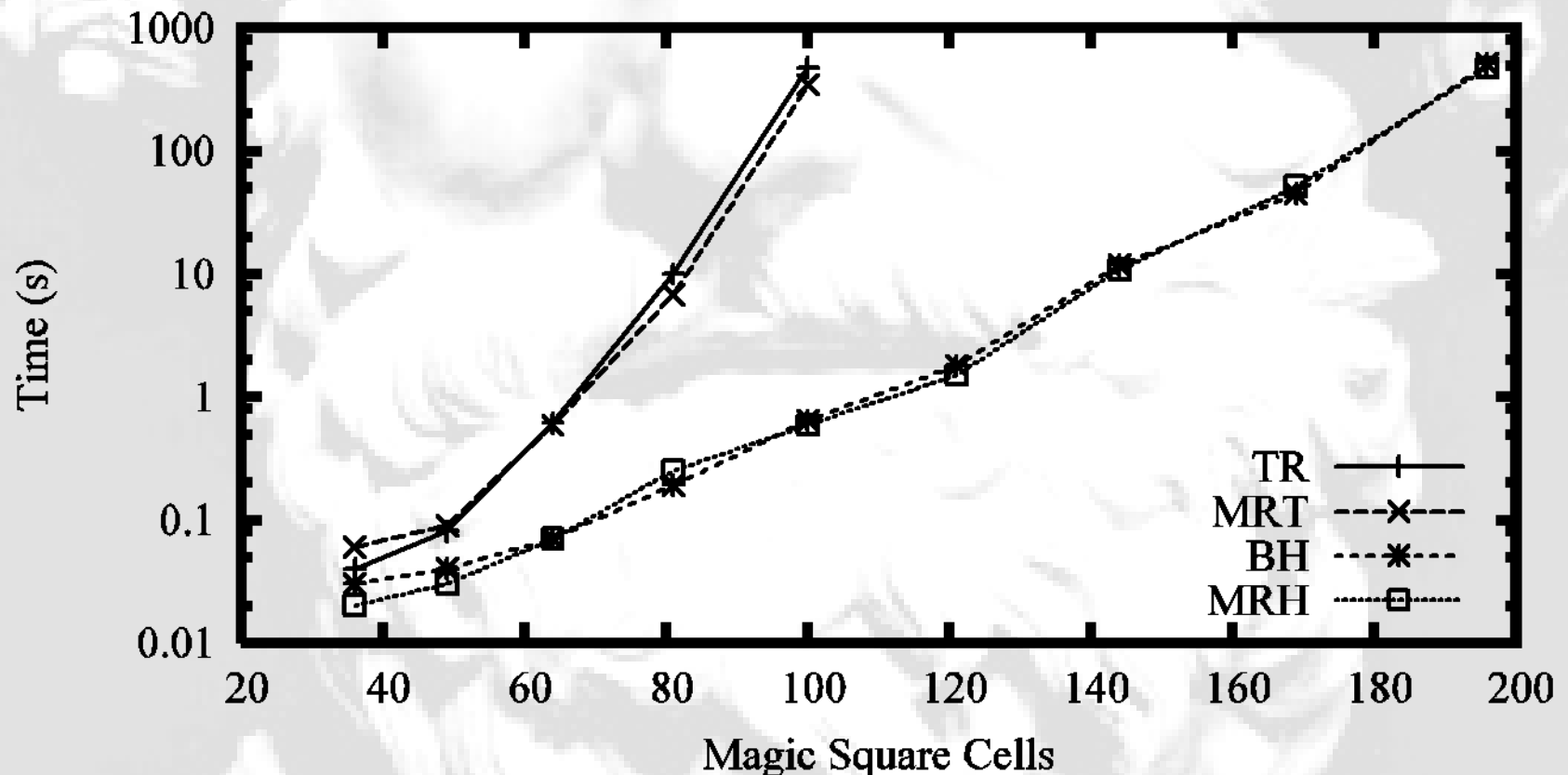
Algorithm

- MRH: bool MetaRestartHybrid (void)
 - InitMoveLimit(maxLocalMoves);
 - while (true) do
 - InitFailLimit(failLimit), InitRandom (heuristic), moves := 0;
 - while (moves++ < maxLocalMoves) do
 - status := TreeSearch(failLimit,heuristic);
 - if (status != inconclusive) then return (status == solved);
 - UpdateHeuristic (heuristic), UpdateFailLimit (failLimit);
 - UpdateMovesLimit(maxLocalMoves)

Diagonally Ordered Magic Squares



Diagonally Ordered Magic Squares



5-SAT

vars	cls	TR-exp	TR-lin	BH-exp	BH-lin
600	7000	29.4	18.1	4.4	2.6
	7050	33	64	8.7	5.7
	7100	46	88	18.7	6.5
	7150	39.5	485	15.6	23
	7200	171	847	92	78

5-SAT

vars	cls	TR-exp	TR-lin	BH-exp	BH-lin
600	7000	29.4	18.1	4.4	2.6
660	7700	67	267	2.3	3.5
720	8400	98	874	9.4	7
780	9100	227	>2K	32	35
840	9800	1062	>2K	103	77

Quasi-Group Completion

order	holes	TR-exp	TR-lin	BH-exp	BH-lin
38	1200	2.5	1.9	0.9	0.9
40		1.4	1.3	0.8	0.7
42		3.3	5.3	0.7	0.8
44		21.3	15.8	5.3	4.1

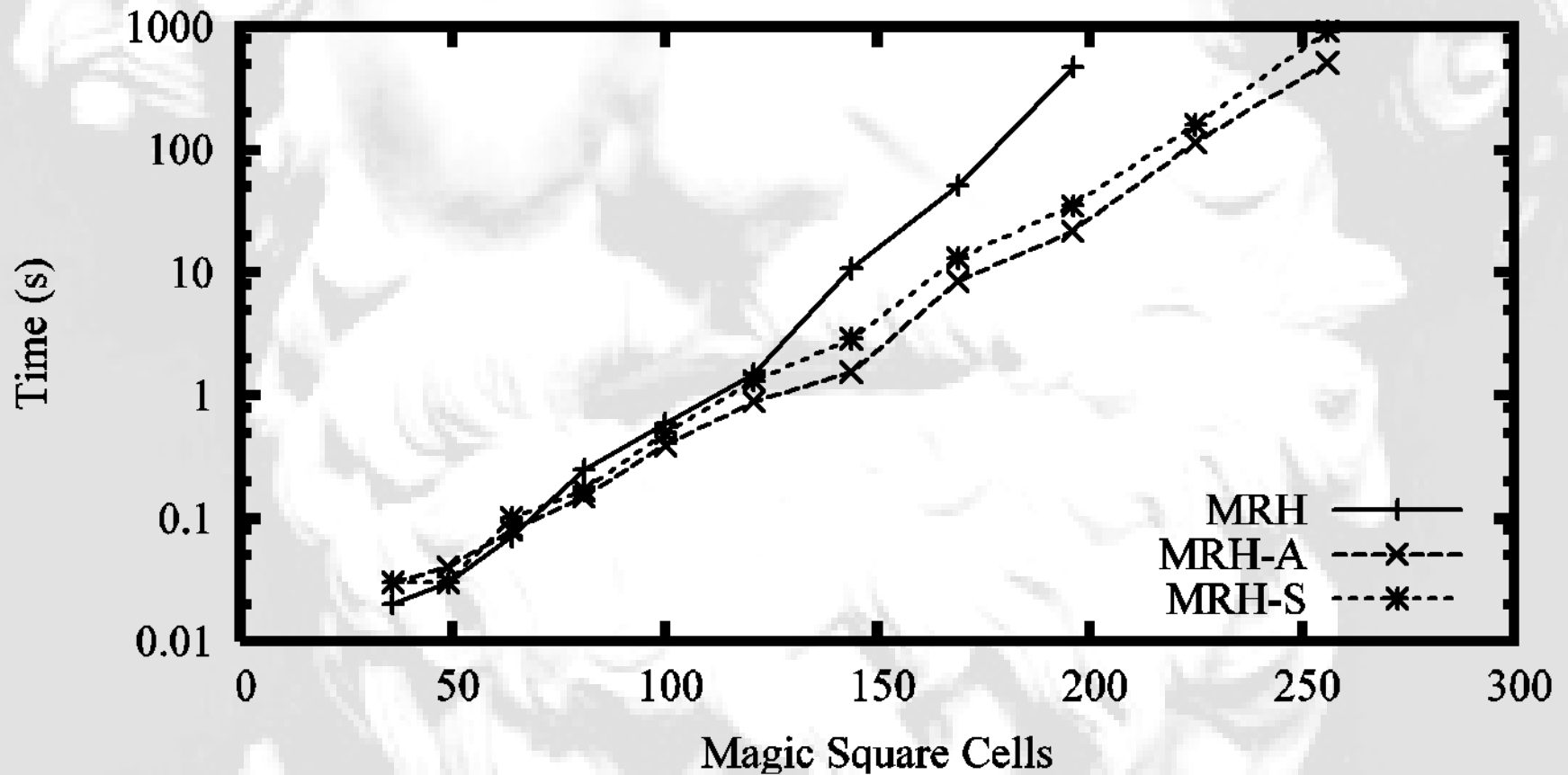
Quasi-Group Completion

order	holes	TR-exp	TR-lin	BH-exp	BH-lin
44	1150	17.4	15.8	4.6	5.0
	1200	21.3	15.8	5.3	4.1
	1250	17.3	23.8	7.2	4.8
	1300	19.5	21.4	4.5	2.8
	1350	16.8	16.5	3.0	1.8
	1400	15.7	31.0	3.2	2.2

Problems With More Than Two Variables

- Heuristic is an ordering of all values in each variable's domain:
 - $HD_1 = (\text{blue}, \text{red}, \text{green})$
 - $HD_2 = (\text{summer}, \text{fall}, \text{spring}, \text{winter})$
- Assume the domains after the last fail are
 - $D_1 = \{\text{red}, \text{green}\}$
 - $D_2 = \{\text{winter}, \text{fall}\}$
- Then we update the heuristic to
 - $HD_1 = (\text{red}, \text{green}, \text{blue})$
 - $HD_2 = (\text{fall}, \text{winter}, \text{summer}, \text{spring})$

Diagonally Ordered Magic Squares



Conclusions

- Local Search should be based on statistical properties found in many instances!
- Robust methods for systematic search space organization are hard to come by.
- Randomization and restarts efficient strategy for search space partitioning.
- However, learning value heuristics works on under-constrained instances!
- Learning value heuristics by coarse-grained local search appears intuitively reasonable.