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,...,f_n] and “distances” (to the nearest optimum) [d_1,...,d_n].
• Let \( \phi_f, \phi_d \) denote the mean and \( \sigma_f, \sigma_d \) the standard deviation of f and d.
• \( c_{fd} := \frac{1}{n} \sum_i (f_i - \phi_f)(d_i - \phi_d) \)
• \( r := \frac{c_{fd}}{\sigma_f \sigma_d} \)
Fitness Distance Correlation

Binary-Code \([r=-.86]\)

Gray-Code \([r=-.57]\)

Meinolf Sellmann

LSCS 2008
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&Tecchiolli ’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

\[ (|x| - 10) \cos(2\pi x) \]

\[ \tilde{V}^\pi(x) \]

\[ V^\pi(x) \]
Learning Evaluation Functions

(a) Run $\pi$ to optimize Obj

(b) produces new training data for $\hat{V}^{\pi}$; retrain the fitter

(c) Hillclimb to optimize $\hat{V}^{\pi}$

d) produces good new starting state for $\pi$
Learning Evaluation Functions

(a) 

(b) 

Vp_1 (second iteration) 

0 30 25 10 

Var(x) = variance 

0 0.05

Meanolf Sellmann
LSCS 2008
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

Meinolf Sellmann

LSCS 2008
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:

Follow Heuristic

Override Heuristic

Propagate, Potentially Override Heuristic

Randomized Variable Order

\[ D_1 = \{t,f\} \quad D_2 = \{t\} \]
\[ D_3 = \{f\} \quad D_4 = \{t,f\} \]
Update of Value Heuristic

• When the fail limit is reached:

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

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

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

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

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

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

![Graph showing the relationship between Magic Square Cells and Restarts for different algorithms: TR, MRT, BH, and MRH. The graph plots the number of restarts on a logarithmic scale against the number of magic square cells.](image-url)
Diagonally Ordered Magic Squares
## 5-SAT

<table>
<thead>
<tr>
<th>vars</th>
<th>cls</th>
<th>TR-exp</th>
<th>TR-lin</th>
<th>BH-exp</th>
<th>BH-lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>7000</td>
<td>29.4</td>
<td>18.1</td>
<td>4.4</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>7050</td>
<td>33</td>
<td>64</td>
<td>8.7</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>7100</td>
<td>46</td>
<td>88</td>
<td>18.7</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>7150</td>
<td>39.5</td>
<td>485</td>
<td>15.6</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>7200</td>
<td>171</td>
<td>847</td>
<td>92</td>
<td>78</td>
</tr>
</tbody>
</table>
## 5-SAT

<table>
<thead>
<tr>
<th>vars</th>
<th>cls</th>
<th>TR-exp</th>
<th>TR-lin</th>
<th>BH-exp</th>
<th>BH-lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>7000</td>
<td>29.4</td>
<td>18.1</td>
<td>4.4</td>
<td>2.6</td>
</tr>
<tr>
<td>660</td>
<td>7700</td>
<td>67</td>
<td>267</td>
<td>2.3</td>
<td>3.5</td>
</tr>
<tr>
<td>720</td>
<td>8400</td>
<td>98</td>
<td>874</td>
<td>9.4</td>
<td>7</td>
</tr>
<tr>
<td>780</td>
<td>9100</td>
<td>227</td>
<td>&gt;2K</td>
<td>32</td>
<td>35</td>
</tr>
<tr>
<td>840</td>
<td>9800</td>
<td>1062</td>
<td>&gt;2K</td>
<td>103</td>
<td>77</td>
</tr>
</tbody>
</table>
### Quasi-Group Completion

<table>
<thead>
<tr>
<th>order</th>
<th>holes</th>
<th>TR-exp</th>
<th>TR-lin</th>
<th>BH-exp</th>
<th>BH-lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td></td>
<td>2.5</td>
<td>1.9</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>40</td>
<td>1200</td>
<td>1.4</td>
<td>1.3</td>
<td>0.8</td>
<td>0.7</td>
</tr>
<tr>
<td>42</td>
<td></td>
<td>3.3</td>
<td>5.3</td>
<td>0.7</td>
<td>0.8</td>
</tr>
<tr>
<td>44</td>
<td></td>
<td>21.3</td>
<td>15.8</td>
<td>5.3</td>
<td>4.1</td>
</tr>
</tbody>
</table>
## Quasi-Group Completion

<table>
<thead>
<tr>
<th>order</th>
<th>holes</th>
<th>TR-exp</th>
<th>TR-lin</th>
<th>BH-exp</th>
<th>BH-lin</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>1150</td>
<td>17.4</td>
<td>15.8</td>
<td>4.6</td>
<td>5.0</td>
</tr>
<tr>
<td></td>
<td>1200</td>
<td>21.3</td>
<td>15.8</td>
<td>5.3</td>
<td>4.1</td>
</tr>
<tr>
<td></td>
<td>1250</td>
<td>17.3</td>
<td>23.8</td>
<td>7.2</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>1300</td>
<td>19.5</td>
<td>21.4</td>
<td>4.5</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>1350</td>
<td>16.8</td>
<td>16.5</td>
<td>3.0</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>1400</td>
<td>15.7</td>
<td>31.0</td>
<td>3.2</td>
<td>2.2</td>
</tr>
</tbody>
</table>
Problems With More Than Two Variables

• Heuristic is an ordering of all values in each variable’s domain:
  – HD$_1$ = (blue, red, green)
  – HD$_2$ = (summer, fall, spring, winter)

• Assume the domains after the last fail are
  – D$_1$ = {red, green}
  – D$_2$ = {winter, fall}

• Then we update the heuristic to
  – HD$_1$ = (red, green, blue)
  – HD$_2$ = (fall, winter, summer, spring)
Diagonally Ordered Magic Squares

![Graph showing the relationship between magic square cells and time (s). The graph includes three lines labeled MRH, MRH-A, and MRH-S, each representing different methods of generating magic squares. The x-axis represents the number of magic square cells, ranging from 0 to 300, and the y-axis represents time in seconds, ranging from 0.001 to 1000.](image-url)
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.