Metaheuristics for the New Millennium

by

Bruce L. Golden
RH Smith School of Business
University of Maryland

Presented at the University of Iowa, March 2003
Focus of Paper

- Introduce two new metaheuristics
  - Search space smoothing
  - Demon algorithms
- Discuss several variants of each
- Illustrate performance using the traveling salesman problem
- Explain why the new metaheuristics work so well
- Point out connections to “old” metaheuristics
  - Simulated annealing
  - Genetic algorithms
Search Space Smoothing: Overview

- Developed by Gu and Huang in 1994

- Applied to a small number of small TSP instances

- In smoothing, we first normalize distances so that they fall between 0 and 1

- At each iteration, the original distances are transformed by the smoothing transformation. Two-opt is applied. The degree of transformation decreases from one iteration to the next, until the original distances re-emerge.
# Search Space Smoothing Algorithm

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Let $d_{ij}$ = the distance from city $i$ to city $j$. Normalize all distances so that $0 \leq d_{ij} \leq 1$. Specify the schedule for the smoothing factor as $\alpha = 6$ (begin), 3, 2, 1 (end).</td>
</tr>
<tr>
<td>2</td>
<td>Generate a random starting tour.</td>
</tr>
<tr>
<td>3</td>
<td>Set $\alpha$ equal to the next value in the smoothing schedule and then smooth the distances according to the following function</td>
</tr>
<tr>
<td></td>
<td>$d_{ij}(\alpha) = \begin{cases} \bar{d} + (d_{ij} - \bar{d})^\alpha, &amp; d_{ij} \geq \bar{d} \ \bar{d} - (\bar{d} - d_{ij})^\alpha, &amp; d_{ij} &lt; \bar{d} \end{cases}$</td>
</tr>
<tr>
<td></td>
<td>where $\bar{d}$ is the average inter-city distance.</td>
</tr>
<tr>
<td>4</td>
<td>Apply a local search heuristic to the TSP with the smoothed distances to produce the current tour.</td>
</tr>
<tr>
<td>5</td>
<td>If $\alpha = 1$, stop. The current tour is the final tour. Otherwise, using the current tour, go to Step 3.</td>
</tr>
</tbody>
</table>
Search Space Smoothing: Summary

- Smoothing clearly outperforms two-opt and takes approximately the same amount of time.

- Smoothing, like simulated annealing, can accept uphill moves.

- Smoothing suggests a new way of classifying heuristics.

- Further experimentation with different “smoothing” functions has led to even better results.
Other Smoothing Functions (Coy et al., 2000)

- Exponential
- Hyperbolic
- Sigmoidal
- Logarithmic
- Concave
- Convex
Part II: Demon Algorithms

- Review previous work
  - simulated annealing (SA)
  - the demon algorithm
  - preliminary computational work
- Introduce three new demon algorithm variants
- Perform a computational study
- Present conclusions
Simulated Annealing

- Generate an initial tour and set $T$ (temperature)
- Repeat until stopping condition:
  - Generate a new tour and calculate $\Delta E$ (change in energy)
  - If $\Delta E \leq 0$, accept new tour
  - Else, if $\text{rand}(0,1) < \exp(-\Delta E/T)$, accept new tour
  - Else, reject new tour
  - Implement annealing schedule ($T = a \times T$)
- The choice of $T$ and $a$ are essential
Demon Algorithms

- Wood and Downs developed several demon algorithms for solving the TSP
- In DA, the demon acts as a creditor
- The demon begins with credit = $D > 0$
- Consider an arc exchange
- If $\Delta E < D$, accept new tour and $D = D - \Delta E$
- Arc exchanges with $\Delta E < 0$ build credit
- Arc exchanges with $\Delta E > 0$ reduce credit
Demon Algorithms (continued)

- To encourage minimization, Wood and Downs propose two techniques
  - Impose an upper bound on the demon value, restricting the demon value after energy decreasing moves
  - Anneal the demon value

- Wood and Downs also propose a random component
  - The demon value is a normal random variable centered around the demon mean value
  - All changes in tour length impact the demon mean value
Demon Algorithms (continued)

- This leads to four algorithms (Wood and Downs)
  
  - Bounded demon algorithm (BD)
  - Randomized bounded demon algorithm (RBD)
  - Annealed demon algorithm (AD)
  - Randomized annealed demon algorithm (RAD)
New Demon Algorithms

- Two new techniques come to mind (Pepper et al.)
  - Annealed bounded demon algorithm (ABD)
  - Randomized annealed bounded demon algorithm (RABD)

- The idea is to impose a bound on the demon value (or demon mean value) and anneal that bound in ABD and RABD

- For RAD and RABD, anneal both the bound on the demon mean and the standard deviation. This leads to two additional algorithms, ADH and ABDH
Computational Study

- Eleven algorithms in all
- We selected 29 instances from TSPLIB
- The instances range in size from 105 to 1,432 nodes
- The instances have different structures
- Each algorithm was applied 25 times to each instance from a randomized greedy start
- Best and average performance and running time statistics were gathered
Preliminary Computational Results & Observations

- Simulated annealing was best overall
- RABD and ABD are nearly competitive with SA
- The intuition behind the hybrids makes sense, but parameter setting becomes more difficult
- The normal distribution can be replaced by “easier” distributions
- Smarter DA variants may exist
Parameter Settings

- We selected three representative test instances
  - For each algorithm, a GA determines a set of parameter values (parameter vector) that works well on these instances

- Resulting parameter vector is applied to all 29 instances
New Variant #1: Triangular Demon Algorithm

- Instead of sampling from a normal distribution, the demon value is sampled from the p.d.f. below.

![Diagram showing a triangular distribution with points at 0.5 D_M, D_M, and 1.5 D_M]
New Variant #2: Uniform Demon Algorithm

- Instead of sampling from a normal distribution, the demon value is sampled from the p.d.f. below

```
DM0.5 DM 1.5 DM

0.5 DM_ DM_ 1.5 DM
```
New Variant #3: Annealed Uniform Demon Algorithm

- Instead of sampling from a normal distribution, the demon value is sampled from the p.d.f. below

- $f$ is set to 0.5 initially and is annealed over time
Advantages of New Variants

- Only two parameters need to be set (initial demon value and annealing schedule) – same as for SA

- The mean and standard deviation are annealed at the same time

- Sampling is easier in these three cases than sampling from a normal distribution
Experimental Design

- We selected 36 symmetric, Euclidean instances from TSPLIB

- The instances range in size from 105 to 1,432 nodes

- For each algorithm, parameters were set using a GA-based procedure on a small subset of the instances
Setting Parameters

- Single-stage genetic algorithm

- Fitness of parameter vector $v$ is

$$F(v) = 100 \sqrt{\frac{1}{m} \sum_{i=1}^{m} ((D(v, i) / B(i)) - 1)^2}$$

where $m$ is the number of test problems in the subset, $D(v, i)$ is the tour length generated by vector $v$ on test problem $i$, and $B(i)$ is the optimal solution to problem $i$

- The fitness is the root mean square of percent above optimal
Experimental Design (continued)

- Starting tour
  - greedy heuristic
  - savings heuristic

- Tour improvement using 2-opt

- Termination condition: 50 iterations of no improvement after going below the initial tour length or a maximum of 500 iterations
Experimental Design (continued)

- Each algorithm was run 25 times for each of the 36 instances

- Averaged results are presented

- All runs were carried out on a Sun Ultra 10 workstation on a Solaris 7 platform

- The six best algorithms are compared
## Experimental Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Greedy Tour</th>
<th></th>
<th>Savings Tour</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average % above optimal</td>
<td>Standard deviation</td>
<td>Running time (hours)</td>
<td>Average % above optimal</td>
</tr>
<tr>
<td>Simulated Annealing</td>
<td>2.85</td>
<td>1.19</td>
<td>17.45</td>
<td>2.84</td>
</tr>
<tr>
<td>Annealed Bounded</td>
<td>2.85</td>
<td>1.34</td>
<td>22.38</td>
<td>2.38</td>
</tr>
<tr>
<td>Randomized Annealed Bounded</td>
<td>3.54</td>
<td>1.53</td>
<td>13.19</td>
<td>3.12</td>
</tr>
<tr>
<td>Uniform</td>
<td>2.86</td>
<td>1.54</td>
<td>24.47</td>
<td>2.67</td>
</tr>
<tr>
<td>Annealed Uniform</td>
<td>2.74</td>
<td>1.28</td>
<td>18.90</td>
<td>2.65</td>
</tr>
<tr>
<td>Triangular</td>
<td>3.14</td>
<td>1.41</td>
<td>20.90</td>
<td>2.51</td>
</tr>
</tbody>
</table>
Part II: Conclusions

- With a greedy start, Annealed Uniform is best

- When the savings heuristic is used at the start
  - Annealed Bounded is best
  - Triangular and Annealed Uniform also perform well and beat SA

- Demon algorithms are sensitive to the starting conditions

- Using the savings heuristic significantly reduces computation times

- Demon algorithms can be applied to other combinatorial optimization problems
Final Comments

- Smoothing and demon algorithms are widely applicable
- They are simple and elegant
- There are few parameters to tune
- They involve approximately 20 lines of code