Using Python to Solve Computationally Hard Problems
Using Python to Solve Computationally Hard Problems

Rachael Madsen

Optimal Design Software LLC

– BS in Mathematics
– Software Engineer & Architect
– Python programmer

rachael@optimaldesignsoftware.com
NP-Complete Problems
NP-Complete Problems

An individual solution can be checked for correctness in polynomial time,
but..

A solution cannot be derived in polynomial time
## NP-Complete Problems

<table>
<thead>
<tr>
<th>Nodes</th>
<th>1</th>
<th>5</th>
<th>10</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial ($n^2$)</td>
<td>1</td>
<td>25</td>
<td>100</td>
<td>4096</td>
</tr>
<tr>
<td>Exponential ($2^n$)</td>
<td>2</td>
<td>32</td>
<td>1024</td>
<td>$1.8 \times 10^{19}$</td>
</tr>
<tr>
<td>Factorial ($n!$)</td>
<td>1</td>
<td>120</td>
<td>$3.6 \times 10^6$</td>
<td>$1.3 \times 10^{89}$</td>
</tr>
</tbody>
</table>
NP-Complete Problems

- Sequencing tasks or objects for any optimization
- Database design & normalization
- Games
The Traveling Salesman Problem
Traveling Salesman Problem

Given a list of cities and their pairwise distances, find the shortest possible route that visits each city exactly once and returns to the origin city.
Algorithms
Changing data may change which algorithm is best
Algorithms

Searching available software implementations may yield the best results
Considerations
Ease of Implementation
Quality of Solution
You are looking for a solution, not the solution
\[ \pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left[ \frac{4}{8k + 1} - \frac{2}{8k + 4} - \frac{1}{8k + 5} - \frac{1}{8k + 6} \right] \]
Will this algorithm generate valleys?
Algorithms > Considerations

Complexity
Algorithms > Considerations > Complexity

• Simple
  – 50 lines of code

• Complex
  – 100,000+ lines of code
Algorithms > Considerations

Parallelization
Algorithms \( \text{> Considerations} \)

**Processing Power**
Starting From Scratch
Starting From Scratch

(it’s a bad idea)
Python Options
Python Options

Where to Look
Python Options > Where to Look

• Google code
• Git
• Forums
• Technical books
Python Options

Licenses
Evaluation
Python Options > Evaluation

Operations Research Tools
Developed at Google

http://code.google.com/p/or-tools/source/browse/trunk/python/tsp.py?r=303
Python Options > Evaluation > or-tools

Runtime

• Lines of code: 132
• Hardware availability
Quality of Solution
Github user trevlovett
Python Ant Colony TSP Solver

https://github.com/trevlovett/
Python-Ant-Colony-TSP-Solver
Runtime

- Lines of code: 132
- Hardware availability
Quality of Solution
Optimization
Optimization

- Confirm implementation of algorithm as written
- Check for most efficient coding practices
Optimization

• Use scientific package such as SciPy to rewrite code for speed

• Break out select code into C routine
Optimization

• Parallelize
  – GPU
  – CPU
Optimization

Beware of fencepost errors and differences in hardware
Web Resources
Web Resources

- **The Stony Brook Code Repository**, Concorde TSP Solver

- **TSP**, github user denlunev, https://github.com/denlunev/TSP

- **Operations Research Tools**, Google,
  http://code.google.com/p/or-tools/source/browse/trunk/python/tsp.py?r=303

- **Ant Colony TSP Solver**, github user trevlovett
  https://github.com/trevlovett/Python-Ant-Colony-TSP-Solver
Bibliography
Bibliography

• *Introduction to Graph Theory*, 2\textsuperscript{nd} Edition, Douglas Brent West, Prentice Hall, 2000


Bibliography


• **Combinatorial Optimization**, Cook, Cunningham, Pulleyblank & Schrijver, Wiley-Interscience, 1997

• **Local Search in Combinatorial Optimization**, Aarts & Lenstra, Wiley, 1997

Bibliography

• Counting: The Art of Enumerative Combinatorics, Martin, Springer, 2001

• Counting and Configurations, Herman, Kucera & Sims, Springer, 2003

Thank You!