Department of Mathematics and Computer Science
University of Southern Denmark, Odense

September 25, 2006
Marco Chiarandini

DM63 - Heuristics for Combinatorial Optimization Problems – Weekly Notes

Lecture 3, Fall 2006

Lecture September 25

We looked at the results of the Competition, Task 1. Only Maciej Balakalarz succeeded in implementing 3 working heuristics for the TSP. The comparison showed that the Nearest Neighbor (NN) heuristic is the best choice among the three submitted. His implementation of NN outperformed Kirsten Falk's implementation in both quality and computation time.

We decided to postpone the launch of the Task 2 to the next Monday in order to allow late submissions and improvements of the current submissions (see Task 1-bis below).

With regard to the previously introduced SAT problem, we discussed instance preprocessing and the phenomenon of Phase Transition. We then introduced the Constraint Satisfaction Problem.

We continued with the analysis of the Local Search components. We stressed the importance of delta evaluations and neighborhood pruning. We also distinguished between best and first improvement rules.

Bibliographical Notes

Comparisons of famous implementations of heuristics for TSP are available at the 8th DIMACS Implementation Challenge on TSP http://www.research.att.com/ dsj/chtsp/index.html.

An alternative paper to the one by Bentley on the TSP is:

> The Traveling Salesman Problem: A Case Study in Local Optimization, D. S. Johnson and L. A. McGeoch, Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra (editors), John-Wiley and Sons, Ltd., 1997, pp. 215-310.

It covers selected tour construction heuristics, classical local optimization algorithms (2-Opt, 3-Opt, Lin-Kernighan), simulated annealing, tabu search, genetic algorithms, and neural net algorithms.

SAT and CSP, and their derived MAX-SAT and MAX-CSP, are treated in Chapters 6 and 7 of Hoos and Stützle book.

The part on Local Search components is also taken from the Hoos and Stützle book from Chapters 1 and 2 through page 66.

The next lecture we will treat GRASP (described at page 89 of the book by Hoos and Stützle), Beam Search (described at page 455 of the book by Hoos and Stützle) and Variable Neighborhood Search (Notes article 3).

Competition – Task 1-bis

This task is a re-edition of Task 1 mainly aimed at allowing those who did not submit their entry within the deadline to catch up the others. The new **deadline** is fixed to **Friday 29, 2006 at 15.00**.

The results from the Task 1 indicated that the heuristics submitted have still margins of improvements. Hence, even those who submitted for Task 1 are invited to check their implementations and resubmit an improved version. Note that it is also possible to change the heuristics with respect to the first submission (this because the results may have suggested some heuristics as more promising than others). Please write in the README file which heuristics are implemented.

Follows the text from Task 1:

Implement basic versions of three construction heuristics for TSP.

One heuristic at choice from the category Heuristics that Grow Fragments

- *Nearest neighborhood heuristics (*`NN`*)*
- *Double-Ended Nearest Neighbor heuristic (*`DNN`*)*
- *Multiple Fragment heuristic (*`greedy`*)*

One heuristic at choice from the category Heuristics that Grow Tours

- *Nearest Addition (*`NA`*)*
- *Farthest Addition (*`FA`*)*
- *Random Addition (*`RA`*)*
- *Nearest Insertion (*`NI`*)*
- *Farthest Insertion (*`FI`*)*
- *Random Insertion (*`RI`*)*
- *Clarke-Wright savings heuristic (*`savings`*)*

One heuristic at choice from the category Heuristics based on Trees

- *Minimum span tree heuristic (*`MST`*)*
- *Christofides' heuristics (*`chris`*)*
- *Fast recursive partitioning heuristic (*`part`*)*

Resolve any tie decision in the construction process by random choices. For details on basic algorithms for graphs use the book "Introduction to Algorithms" by Cormen et al.

The test instances and the code implementing a framework for construction heuristics and local search algorithms are available from the Course Section Resources. The code contains the functions for reading the instances.
Read the documentation under the Course Section Competition for the modalities of submission. The program must run with an additional option: `-c NAME_HEURISTIC`*. The* `NAME_HEURISTIC` *for each construction heuristic is given between parentheses above. Write in the README file which heuristics have been implemented.*