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

October 13, 2006
Marco Chiarandini

DM63 - Heuristics for Combinatorial Optimization Problems – Lecture Notes

Lecture 8, Fall 2006

Lecture October 12

We revised Iterated Local Search from the slides by Stützle which match the content of article 5 in the Notes.

We commented the results from the Task 2 of the competition. The Task had a clear winner, Brian Kristensen, whose entry compared similarly to the algorithm by Stützle. Emphasis was given on the presentation of the experimental results which may be graphical or textual. In the former case we pointed out boxplots, interaction plots, and scatter plots as useful means to convey information. In the latter case numerical tables must be adequately commented and important numbers emphasized.
Tools for producing graphics are available in any package for statistical analysis under Windows, MacOs and Linux. In particular the use of the open-source project R (http://www.r-project.org/) has been recommended. Finally examples have been shown of program profiling.
These kinds of analysis should be included in the exam project.

As class exercise we treated the Set Covering Problem. We devised local searches and compared with the similar problems of partitioning and packing.

The exercise served us to introduce the methods Iterated Greedy, Squeaky Wheel Optimization and Multilevel Refinement. Finally we discussed the general framework for the use of Very Large Scale Neighborhoods. There is no mention of these methods on the Notes. The information provided in class is fairly enough for this course. However for those interested below are some references.

There are no lectures in Week 42. The next lecture is October 23 when we will treat Genetic Algorithms. The lecture after that we will treat Ant Colony Optimization. These methods are described in the Notes, article 7 and 8, respectively.

During the pause students can work at the Task 3 of the competition described below.

Bibliographical Notes

Iterated Local Search is described in article 6 of the Notes.

The Set Covering Problem (SCP) is treated in the book by Hoos and Stützle from page 488. A description of Iterated Greedy and application to the SCP is also given in those pages.

An article on Squeaky Wheel Optimization is:

> David Joslin, David P. Clements: Squeaky Wheel Optimization. J. Artif. Intell. Res. (JAIR) 10: 353-373 (1999)

An article on Multilevel Refinement is:

> C. Walshaw. Multilevel Refinement for Combinatorial Optimisation Problems. Annals Oper. Res., 131:325-372, 2004.

A survey on Very Large Scale Neighborhoods is:

> R.K. Ahuja, O. Ergun, J.B. Orlin, and A.P. Punnen. 2002. A survey of very large scale neighborhood search techniques. Discrete Applied Mathematics 123, 75-102.

Competition Task 3

The submission deadline is at 23.59 of Wednesday, October 25.

Implement and configure one of the following metaheuristics. There is freedom in the choice of components. Those in parenthesis are however some suggestions.

- Simulated Annealing [`SA`] (*initial solution*: random or nearest neighbor heuristic; *neighborhood*: 2-opt in its basic version; *proposed step*: one at random; *acceptance criterion*: metropolis; *initial temperature*: such that 3-4% of proposed steps are not accepted; *cooling scheduling*: geometric, ie, $T_{i+1} = \alpha T_i$, $\alpha \approx 0.95$; *temperature length*: $|V| \cdot (|V| - 1)$).
- Guided Local Search [`GA`] (*local search*: the most efficient 2-opt of those implemented for Task 2; *solution components*: the edges and the cost contribution is the edge length; *update penalties*: the length of the edges of maximal utility are updated by a factor $\lambda = 0.3\bar{s}/n$ where $\bar{s}$ is the average length of good tours).
- Iterated Local Search [`ILS`] (*initial solution*: random or nearest neighbor heuristic; *local search*: the most efficient 2-opt of those implemented for Task 2; *perturbation*: double bridge; *acceptance criterion*: always the current solution)

Beside the examples given in class, applications of these metaheuristics to the TSP are also described in the book by Hoos and Stützle on pages 72 through 88.

You are allowed to enter one only final algorithm. Values of parameters must be embedded in the program. The program must be launched with the following command line:

```
tsp -i name-instance -m name-algorithm -s seed -t time-limit
```

where `name-algorithm` is one of the labels associated to the metaheuristics in the list above (write in the README file which). The output format remains the same as in the previous tasks. Note that the time-limit will be used hence make sure that the program stops when the limit is exceeded and returns the best ever solution.

The comparison will be based only on the final solution quality, as all the programs will be run using equivalent time limits.

In order to determine a fair time limit among different implementations and programming languages a *Random Restart Iterative Improvement* (`RRII`) heuristic must also be implemented and delivered. In this case the program must run with

```
tsp -i name-instance -m RRII -s seed
```

RRII must perform **200** restarts. Every restart comprises the construction of a solution by means of the Nearest Neighborhood heuristic (`NN` from Task 1) and the application of the local search: 2-opt first improvement fixed radius nearest neighbor search (`FFR`, implemented in Task 2).

On each instance i, RRII will be run for 5 times and the CPU time of each of the 5 runs will be measured. The time limit will then be computed as the average value $\bar{t}_i$ over the 5 runs. The time limit will therefore differ over the instances.
This time limit can be computed by the participant as well and used to configure, tune and compare different variants of the metaheuristic developed. To speed up this phase a new set of smaller instances has been made available at http://www.imada.sdu.dk/ marco/DM63/Material/.

Exercises

Exercise 1

Make clear the meaning and differences of the following pairs of concepts

- problem vs instance
- approximation algorithms vs approximate algorithm
- objective function vs evaluation function
- efficiency vs effectiveness

Exercise 2

Recall one or more construction heuristic for the following problems encountered in the lectures.

- satisfiability
- constraint satisfaction
- graph coloring
- traveling salesman
- single machine total weighted tardiness
- p-median
- quadratic assignment
- set covering

Exercise 3

For the same problems at Exercise 2 define one or more promising candidate solution representations and neighborhood structures.

Exercise 4

Recall the main idea and the algorithmic sketch of the following Metaheuristics treated in the lectures:

- beam search
- greedy randomized adaptive search procedure
- variable neighborhood search (and extensions)
- variable depth search (Lin-Kernighan heuristic for TSP)
- ejection chains and dynasearch
- very large scale neighborhoods
- random restart iterative improvement
- randomized iterative improvement (min-conflict heuristic, novelty)
- probabilistic iterative improvement (metropolis algorithm)
- simulated annealing (and noising method, threshold method, old bachelor acceptance)
- tabu search

- dynamic local search (guided local search)
- iterated local search
- iterated greedy
- multilevel refinement

Exercise 5

Imagine to have a black box algorithm implementing a local search and that this black box cannot be open, that is, the algorithm cannot be modified. Which among the metaheuristics mentioned in Exercise 4 can be used to improve the results of the black box local search? In other terms, which metaheuristics can use a subsidiary local search algorithm without having to modify it? (The black box accepts as inputs: a problem instance, an initial solution, a function to evaluate the solutions; and outputs a final solution. No neighborhood structure and search space can be passed).

Exercise 6

Sometime problems can be transformed into others problems for which powerful solution methods are known. This is the case of the decision version of the graph coloring which can be encoded as a satisfiability problem. Show how this can be achieved by defining the appropriate literal variables and the clauses that correspond to the edge constraints in the graph coloring. It is often the case that such kind of transformations let grow considerably the search space and are therefore not leading to an efficient solution approach. In the case under analysis however, showing that the transformation from GCP and SAT can be done in polynomial have an important implication on the definition of the complexity of the GCP. Which is this implication?

Exercise 7

A particular case of the Graph Partitioning problem consists in finding a partitioning of a graph $G = (V, E)$ into two subsets of vertices V_1 and V_2 with $|V_1| = |V_2|$ such that the number of edges connecting vertices from V_1 and V_2 is minimized.

- Design one or more construction heuristics
- Define one or more solution representations and neighborhood structures
- The Variable Depth Search method implements a way to search an exponentially large neighborhood in polynomial time. This is made possible by some intelligent pruning of the search and heuristic rules. Devise a Variable Depth Search algorithm for the Graph Partitioning problem.
  (Hint: find the best pair $x_1 \in V_1$ and $y_1 \in V_2$ to swap. Swap these vertices and fix their position, that is, they cannot be swapped again in the step. In the kth iteration, $1 \leq k \leq n$, find the best pair x_k, y_k among the remaining $(n-k)^2$ pairs, swap this pair and fix the position of x_k, y_k. These may not be swapped again. If k reaches n then the roles of V_1 and V_2 are simply interchanged. It is important then to stop when $k < n$ swaps have been found such that the improvement in the quality of the partition is maximized. How can this be done efficiently?)