Institut for Matematik & Datalogi
Syddansk Universitet

19. september 2005
JBJ

DM63 Meta-heuristics — Ugeseddel 4

Lecture on September 21

Note that we use the wednesday slot instead of the thursday slot in this week!

I lecture on Tabu-search and iterated local search. Notes pages 20-28 and 163-175 (the last part will not be covered in the lecture, but serves as inspiration and you should read it at some point during the course.) For iterated local search I will use slides. I will put a link to slides on iterated local search on the home page of the course.

Week 39

There are no classes. Instead you should spend your time implementing the (meta)-heuristics we have covered so far and try these out on the graph partitioning problem.

Plan for October 6, 2005

I will lecture on genetic algorithms pages 29-35 in the notes. English speaking participants (if any) see the corresponding chapter in the book by Reeves in the library. We will discuss the pages 81-107 and 163-175 in the notes. You should read this material before the lecture!

Exercises:

1. Implement a raw tabu search algorithm for graph partitioning including procedures for each of the following
   (a) A method for maintaining a tabu list, that is, inserting and deleting elements as well as checking whether a transition (to a neighbour) is tabu.
   (b) A method for calculating the best solution in $N(s)$ for a given solution s.
   (c) A method for finding the best non-tabu neighbour of s.
   (d) A way of graphically representing the intermediate solutions encountered by the algorithm. That is, a plot of the current best solution ala what you did for simulated annealing (see weekly note 2).
2. Experiment on various input data in order to investigate the following
   (a) Implement a steepest decent algorithm and test this on the same data as you use for TS. Save the results so that you can compare them with the ones you obtain for TS.
   (b) Experiment with two variants of TS:
      - The one where $N(s)$ is all (including non-balanced) partitions that can be obtained from s by moving one vertex to the other side and the tabu list simply contains the names of the vertices that may not be involved in a move at the current time.
      - The one where we always maintain a feasible partition ($|X| = n/2$) and we swap pairs of vertices one from X and one from the other set. Here there are several possible choices for the tabu list, based on whether only the swap x, y is considered tabu after swapping x and y or all swaps involving one of x, y is considered tabu. **Try both and compare your results**.

(c) For a fixed choice of $N(s)$ (its structure) investigate the influence of the length of the tabu list on the quality of the solutions obtained by the algorithm.

(d) For a fixed choice of $N(s)$ and structure of the tabu list investigate the influence of various aspiration criteria on the quality of the solutions. Test this for at least the following two criteria:

- Choose a solution $s' \in N(s)$ even if it is currently tabu provided that s' is the overall best solution seen so far.
- Choose a solution $s' \in N(s)$ even if it is currently tabu provided that s' is better than the earlier solution which has caused s' to be tabu at this time (what this means precisely depends on your definition of being tabu).

(e) You should also try the following idea which is called path relinking:

- During the execution of the algorithm we maintain a set of 5-10 very good (so called elite) solutions among those we have seen so far.
- Then when the algorithm gets stuck in a local optimum one tries to form a "path" from the current solution s to one of the elite solutions. That is, we determine a set of swaps that will transform s into one of the elite solutions (this is easy since we know the partitions for s and each elite solution). In fact we can easily find that solution s' among the elite solutions which is furthest away from s in terms of the number of swaps needed to go from s to s' .
- Suppose now that we have found such a path P from s to s' (consisting of a number of swaps, each corresponding to one edge of P). Partition P into suitable pieces (say 5 swaps per piece) and perform a normal steepest decent from each end of such a piece. Record the best solution found in this process and if it is better than the current best (before we did path relinking step) this is updated and similarly the elite solutions are updated if we find better ones during the process.

The philosophy behind path relinking is that on such a "path" from a local optimum to an elite solution there could be one or more valleys with very good solutions.

3. Implement a simple Iterated local search algorithm for the graph partitioning problem.

(a) Experiment with various ways of permuting the solution when we reach a local optimum in descent. One simple strategy is to swap a number of elements between the two sides. Thus you can vary this number and see what is best.

(b) Experiment with various criterias for accepting the new local optimum (as the starting point for the next iteration starting with a permutation). The simplest one is to accept the new local optimum only if it is better than the one we came from. Try to come up with others and compare the results.

(c) Experiment with various termination criteria. A typical one is some number k of rounds with new improvement of the best solution ever found.

(d) You should also try using the version of descent where you only move a vertex to the other side, rather than swap two vertices.