DM63 Final project

Posed by Jørgen Bang-Jensen, Department of Mathematics and Computer Science
University of Southern Denmark

The project will be available from the DM63 course page on October 27'th. 2005. You must hand in your report by noon Monday December 19'th in JBJ's mailbox in the secretary's office.

1 Formalia

The projects are evaluated according to the 13-point scale with external examination. Note that it is not permitted to collaborate with each other concerning the project.

It is very important that you explain how your have obtained your results and how you reach your conclusions based on those results. This includes an explanation of how you have determined the final choice of parameters for the various heuristics. You may hand in your report in English or Danish. **If you would like a receipt showing that you have handed in the report in time, you must ask the secretary for one!**

In order to allow me to notify you in case there is a correction to the project-description or similar, you should send an email to jbj@imada.sdu.dk informing me that you are working on the project. This way I will be able to inform you by email if there are changes.

2 Project description

In the course we have studied the graph partitioning problem (GP) extensively. The **digraph partitioning problem** (DP) is the following obvious generalization of GP: Given a digraph $D = (V, A)$ and a weight function ω on A. Find a partitioning of V into two sets X, Y of equal or almost equal size (if $|V|$ is odd we take X to be the smaller of X and Y), such that the total weight of the arcs from X to Y is minimized. That is we want to minimize

$$dp(X, Y) = \sum_{i \in X, j \in Y} \omega(ij)$$

Just as for the GP problem there are (at least) two possible neighbourhoods that can be used in metaheuristics:

1. The **interchange** neighbourhood where a neighbour is obtained by interchanging a vertex from X with one in Y
2. The **move** neighbourhood where a neighbour is obtained by moving a vertex from X to Y or from Y to X. In the later case we must take care of the cost of getting back to a(n almost) balanced partition.

The so-called **Feedback arc set problem** (FAS) is as follows: Given a digraph $D = (V, A)$, where $V = \{1, 2, \ldots, n\}$ and a real-valued weight function ω on A. Find a permutation $\pi_1, \pi_2, \ldots, \pi_n$ of V (that is, an ordering of V) such that the total weight of those arcs $\pi_j\pi_i$ where $j > i$ (that is, the arcs that point backwards in the ordering) is minimized. Hence we want to minimize

$$fas(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \omega(\pi_j\pi_i)$$

Again there are two obvious neighbourhoods:

1. **interchange** where we interchange two elements in the current permutation.
2. **move** where we select a vertex and a position where we will insert this. Thus if the i'th element of π is selected a move is the operation that inserts π_i anywhere in the remaining permutation (example: if we make a move of 3 in 12345 we can get any of the following neighbours 31245, 13245, 12345, 12435, 12453).

A closely related problem to FAS is the **linear ordering problem** (LO). Here we are given a non-symmetric $n \times n$ matrix M of real numbers and the goal is to permute rows and columns with the same permutation π so as to maximize the sum of the elements in the upper triangular part of M, that is we want to maximize

$$lo(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} M_{\pi_i,\pi_j}.$$

By thinking of the matrix as the weighted adjacency matrix of a digraph, it is easy to see that this problem is equivalent to solving the FAS with the same weight function, since minimizing the weight of backward arcs is the same as maximizing the weight of the forward arcs. Note that there might be loops in the digraph (corresponding to diagonal elements of M), but they play no role in the problems.

There are several construction heuristics for the FAS. One of these is the one we could call **minimum relative in-weight order** (MIO) [1, page 370]: It starts by removing the vertex v for which the sum of the in-weights divided by the sume of the out-weights, i.e.

$$w(i) = \frac{\sum_{k=1}^{n} \omega(ki)}{\sum_{k=1}^{n} \omega(ik)}$$

is the smallest possible. This will be vertex 1 in the ordering. Vertex v (and all arcs incident with it) is removed from the digraph. In the i'th step we choose the vertex whose w value is smallest possible among the remaining vertices (and in the remaining digraph) and let this vertex be number i in the ordering.

One can also construct a heuristic for FAS based on digraph partition. We call this the **recursive split heuristic** (RS). Below DP(G, P) is a (meta)-heuristic for weighted digraph partitioning (P denotes a subset of V and when $P \neq V$ we only consider the subdigraph induced by P). It will return a(n almost) balanced partition X, Y of P with "small" weight on the arcs from X to Y. Then RS(G, P, k) can be described as the following recursive algorithm. Here P is a subset of V and k is an integer in $\{2, \ldots, n\}$.

RS(G, P, k)

1. If $|P| < k$ then solve FAS using a construction heuristic and let Q be the resulting order.
2. Else
   - $T, S := DP(G, P)$ # Makes $T = X$ and $S = Y$ where $X, Y = \text{DP}(G, P)$
   - $T := RS(G, T, k)$ # Recursively order T
   - $S := RS(G, S, k)$ # Recursively order S
   - $Q = S + T$ # concatenate T after S
3. Return Q and value of ordering

In RS you may use any (meta)-heuristic for DP and any construction heuristic for the small cases (including solving optimally).

During the semester you have encountered, among others, the following meta-heuristics and we have discussed how to adapt these to the graph partitioning problem and also FAS:

- Simulated Annealing (SA).
- Genetic Algorithms (GA).
- Tabu Search (TS).
- Iterated local search(ILS)
- The Noising Method (NM).
- The modified noising method with moves and exchanges (MNME) as described in [3, Pages 114-126].
- Guided local search (GLS) including Fast local search (FLS).

You should consult the weekly notes 1-6 for relevant parameters for the various heuristics. see also [2]. We have also discussed the so-called Lin-Kernighan heuristic (LK) for graph partitioning (see weekly note 1).

3 What should you do?

The aim of the project is to perform a detailed comparison of metaheuristics for FAS based on the interchange or move neighbourhoods with the recursive split heuristic RS for FAS (as defined above). The comparison should be based on two different kinds of input:

1. Digraphs with no weight on the arcs. Here the aim is just to minimize the number of arcs backwards in the ordering.
2. Linear ordering instances from the literature. Here the aim is to minimize the total weight of the backwards arcs.

You are expected to address all sub tasks below. Of course, you are not required to hand in a full masters thesis, but you must show that you are able to plan reasonable experiments and argue for your choices based on your findings from these experiments. **Remember: It is not the number of experiments that you make which is the issue of importance, but it is the way you perform your experiments (including what to test next) and your conclusions based on the experiments that is important!**

In order to make it easier for others to read and understand your results you must present them in the form of plots and tables (e.g. using gnuplot, excel or similar). The important thing is that it is easy to see how you reach your conclusions based on the results you obtain. The goal of the project is a report which compares the various methods and hence the reader must be able to see what your conclusions are based on. When referring to a table to conclude something, you must say where in the table the reader can find the things that support your conclusion.

1. You must implement two meta-heuristics for FAS. One of these should be iterated local search (see the notes on the course home page and [1]) and the other can be chosen freely among SA,TS,GA,GLS,NM,NMME. Below $X = ILS$ and let Y denote the other metaheuristic you have chosen.

2. Implement each of the heuristics X and Y. Show a plot of a typical run (using a typical choice of parameters) of the heuristic by plotting the solution value against the number of iterations.

3. For X and Y you must perform and document experiments which illustrate how you tune your heuristics to find good solutions. These tests should be based on your own test data as well as test data from LOLIB and XLOLIB (see below). Your report must contain a discussion of how to tune the algorithms (here you may use the suggestions on the weekly notes concerning what experiments to make for the various heuristics). It is very important that you argue for your final choice of parameters based on the observed results. The discussion should be accompanied by figures which show the quality of the algorithm as a function of the various parameters (compare with [2]).

   In order to obtain reasonably significant results it is important that you experiment of fairly large graphs. Besides answering the relevant questions from the weekly notes your discussion should preferably answer the following questions, unless your choice of X or Y makes the test irrelevant (see also the relevant weekly notes for each heuristic!):

   - How should one perturb a local minimum in ILS? Is it better to use some (meta)heuristic to make the perturbation than to make a certain number of interchanges?
   - If you use interchanges as a perturbation, then how many do you need to make? How does the number depend on the number of vertices?
   - Should one only accept a new local minimum in ILS if it is better than the current one, or is it better to use another criteria.
   - Which neighbourhood is better for FAS, the interchange or the move neighbourhood?
   - Is it good to mutate often in GA?
   - What is a good crossover strategy for GA?

- Should the length of the Tabu list be dependent on n?
- Should the noise be the same for all edges in NM or is it better to noise each edge independently.
- What happens if we just add noise to the objective function rather than to the data?
- What are good features to penalize in GLS?
- How do you implement FLS to make it faster than a standard local search?

4. Run your tuned versions of each algorithm on several of the test graphs from LOLIB and XLOLIB. Compare your results for the LOLIB instances to the optimal solutions listed on the web page
   http://www.iwr.uni-heidelberg.de/groups/comopt/software/LOLIB/
   For the XLOLIB problems I will list best known solutions on the course page as soon as I get them from the authors.

   When you make this comparison you should remember to convert your objective function to the one for the LO problem. This is done by calculating $lo(\pi)$ for the permutation π found by your FAS algorithm

5. Implement and tune a meta-heuristic W of your own choice for the digraph partitioning problem. **You do not have to give a detailed documentation of the tuning**. You can check its quality on the standard GP problems from Johnsons test base and on homemade digraphs where you know the answer. Document this in a table, showing the quality on selected Johnson graphs.

6. Use W and a construction heuristic to implement the RS heuristic for FAS. Below Z denotes the RS heuristic obtained this way.

7. Experiment with the parameter k in Z to see at what stage it is better to shift to the construction heuristic.

8. Perform experiments in order to compare the tuned heuristics X, Y and Z regarding their ability to find good solutions for FAS problems. This comparison should be based on both unweighted digraphs as well as graphs from (X)LOLIB in order to determine whether it is the same heuristic which is the best for all types of test graphs. In these test you must address the following:

   (a) Which heuristic finds the best solution? Is it always the same heuristic?

   (b) How large is the difference in speed among the heuristics. That is, how much time does it take to perform one run of each algorithm (here it is assumed that you use a stopping criterion for each algorithm).

   (c) Suppose now that you only have a certain amount of time available, say 5 minutes and you allow each heuristic precisely this amount of time (by restarting if it finishes early). Which heuristic is now the winner and how does the relative ranking among the heuristics change?

   (d) For those of your heuristics where it makes sense, try the variant where you stop the algorithm earlier and the perform one descent at the end. Does this lead to better results?

9. **Try to explain what you see**. I.e. if one heuristic is much better (much worse) than the others, try to give a reason why this could be so.

4 Test data

In the experiments above you should use

- Your own test data, e.g. graphs where you know a good ordering/partition. This can be obtained in several ways and you should suggest some and try them.
- Benchmark problems from LOLIB:
  http://www.iwr.uni-heidelberg.de/groups/comopt/software/LOLIB/
- Benchmark problems from the XLOLIB:
  http://www.intellektik.informatik.tu-darmstadt.de/ schiavin/lop/
  These files will be made available to you via the homepage of the course
- Methods for making your own test instances:
  1. Make random orientations of graphs from Johnson's test base. The graphs here are undirected so $M_{ij} = 1$ if and only if $M_{ji} = 1$. Delete one of these 1's randomly for each pair i, j with an edge between then.
  2. Make random weighted digraphs from Johnson's graphs. This time you change the weights of the arcs to a random number and do it so that $\omega(ij)$ is not necessarily the same as $\omega(ji)$.
  3. Construct some acyclic (unweighted or weighted) digraphs (e.g. by fixing an order of an undirected graph and orienting all arcs forward).
  4. Combine two digraphs G, H for which you know good orderings by putting a few (or no) arcs between them.
- You are also welcome to use other benchmark sets that you have found on the WWW Remember to state where you have the data from!

References

[1] T. Schiavinotto and T. Stützle, The linear ordering problem: Instances, Search Space Analysis and Algorithms, *Jounrnal of Mathematical Modelleing and Algorithms* **3** (2004) 367-402.

[2] D. S. Johnson, C. R. Aragon, L. A. McGeoch and C. Schevon, Optimization by simulated annealing: an experimental evaluation; part I, Graph partitioning",*Operations research* **37** (1989) 865-892.

[3] Noter til DM63, Efterrssemestret 2005, IMADA Syddansk Universitet.