Evolutionary Computation
Practical Assignment 2

1 MLS, ILS, and GLS for Graph Bipartitioning

The task of this practical assignment is to implement and experimentally compare multi-start local search (MLS), iterated local search (ILS), and genetic local search (GLS) for a graph-bipartitioning (GP) problem. The goal is to divide the set of vertices of a graph in two equally sized subsets such that the number of edges that connect two vertices belonging to two different subsets is minimized.

Local search algorithms iteratively change a solution until no better solution is found in the neighborhood of the current solution. The local search algorithm used is the Fiduccia-Mattheyses (FM) heuristic. MLS, ILS, and GLS are metaheuristic algorithms that improve the performance of the local search algorithm.

1. *Multi-start local search* simply restarts local search from a set of randomly generated initial solutions. The best local optimum found is returned as final solution.
2. *Iterated local search* mutates the current solution found by local search and applies local search from the mutated solution. When the new local optimum is better than the current one, ILS continues its search from this new local optimum, else it simply returns to the previous local optimum. The size of the mutation is a crucial factor in ILS. If the mutation is too small, the local search will return to the local optimum it has just mutated. If the mutation is too large there will be no correlation between the new local optimum and the one that was mutated. ILS is then reduced to a MLS algorithm.
3. *Genetic local search* applies selection and recombination to a population of local optimal solutions obtained by a local search algorithm, in this case the FM heuristic. After applying recombination, the offspring is optimized with local search to become a new local optimum. The solutions in the population of GLS are thus all local optimal solutions. Selection focuses the search towards good regions in the search space, while recombining local optima generates good starting points for the local search algorithm (= FM) in order to find new local optima.

2 Experimental study

Set up an experimental study that investigates the following algorithms. Note that to experimentally compare two algorithm instances, you need to run the algorithms 10 times for each specific choice of parameters and compare the median results of these runs with each other. Use a statistical significance test (preferably the Wilcoxon-Mann-Whitney aka. Mann-Whitney U test) to check whether the observed differences are indeed statistically significant. Box plots are a useful way to show results (see lecture slides). Also report the best value found for the 10 runs (and how many times out of 10 this best value was found).

1. Implement MLS with the FM local search. In one pass of the FM heuristic all nodes are swapped to the other partitioning in a greedy way. We compare all algorithms here based on the same number of FM passes, specifically 10000 passes. Measure and report also the computational time required.

2. Investigate the impact of different mutation/perturbation sizes for ILS with the FM local search. Start with the smallest possible. Keep investigating larger perturbation sizes until the performance of ILS drops. Measure the average amount of times that the newly obtained local optimum is equal to the local optimum before perturbation (meaning that the search has not left the region of attraction of the perturbed local optimum). Are the ILS versions statistically better/worse than MLS ? Are the results obtained with the different mutation/perturbation sizes statistically different from each other ?

3. Implement and test GLS (using the FM local search) with population size 50. The specific genetic algorithm is an incremental (or steady state) GA where there is no explicit notion of generations: each iteration two parents are randomly selected, use uniform crossover to generate one child, do FM local search on the child, let this optimized child compete with the worst solution in the population, if it is better or equal it replaces the worst solution. Are the GLS results statistically better/worse than MLS or ILS ?

4. Select your own topic to investigate. Clearly specify the research goal and discuss the results. Possible topics might be the impact of population size, increasing the selection pressure, combining ILS with GLS by adding mutation/perturbation (how to combine them ?), use more (or less) FM passes than 10000, apply an adaptive mutation/perturbation step size in ILS, compare with another metaheuristic, for instance, simulated annealing, ...

   Compare the above algorithms in two ways:

   (a) Compare them on the basis of generating the same number of FM passes (= 10000),

   (b) Compare them on the basis of the same run time (= physical clock time). Measure the time needed for a single MLS run (= 10000 FM passes), and let ILS (with the best performing perturbation size you have found previously) run for the same amount of time. Do the same for the GLS - this is, one single run. Do this 25 times (for a single MLS, ILS, and GLS run) and tabulate the means and plot a boxplot. Note that the run time should not be exactly the same number, you can let a FM algorithm finish its current pass before stopping it whenever it has reached the time limit.

3 Graph Bipartitioning

- The experiments are tested on a single planar graph of 500 vertices (see file Graph500.txt). The graph is specified in a text file with each line giving the ID number of the vertex, its coordinates in the plane, the number of connected vertices, the ID numbers of the connected vertices. For instance the line:
  1 (0.502987,0.528829)   8   28 102 162 233 360 393 460 500
  states that vertex 1 is connected to 8 other vertices with indices 28 102 162 233 360 393 460 500. (note: you do not need the coordinates for this assignment). Be careful not to read the 8 in the above example as "vertex 1 is connected to vertex 8"!

- Solutions to the GP problem are represented by binary strings $x_1 \ldots x_\ell$ with ℓ the number of nodes in the graph and $x_i \in \{0, 1\}$ specifying to which of the two partitions the node i belongs. Note that the number of ones and zeroes in each solution string needs to be equal to $\ell/2$.

- Recombination is done by uniform crossover that respects the equality constraint between the number of ones and zeroes. Remove the redundancy in the representation as follows:

  1. Compute the hamming distance between the two parents. If the distance is larger than $\ell/2$ invert all bit values of one parent.
  2. Generate a single child that has the same bit value as the parents whenever the two parents have the same value for that vertex.
  3. The positions in the child string where the two parents disagree are randomly filled in under the constraint that the total number of ones and zeroes in each solution needs to be equal to $\ell/2$.

- It is important to implement the FM local search algorithm efficiently. Moving a vertex with maximum gain to the other partitioning and updating the gains of its connected vertices should be done in constant time. You can create your own implementation, but one way to achieve this is to keep the gain information in two (implicit) double linked list. Each vertex i can be put in an array of size 500 at index *i-1* Each vertex has a list of its connected vertices, its gain value, the index of its predecessor vertex in the linked gain list, and the index of its successor vertex in the linked gain list. The two gain lists are arrays of size (2.MaxDegree + 1). The index of the first vertex in the linked list with a gain value k is kept at the corresponding index in the gain array. A pointer keeps track of the maximum gain in both gain arrays. When a vertex is moved to another partitioning - and thus removed form the gain array - the gain of its connected vertices and there membership of the linked gain list needs be updated: this can be done in constant time by updating the predecessor and successor values of the transferred vertex and all its connected vertices. When building the gain linked lists anew after FM has passed through the entire graph, you should rebuild the linked lists in random order (you do not want FM to traverse the linked lists in exactly the same order as during the first pass of FM through the graph).

- You may use a Large Language Model for Coding to assist you in the creation of your program but you should explain in the report what the LLM suggested and how you integrated that in your implementation.

4 Report

Write a report discussing your results, the observations, and the conclusions you can draw. Also give a short description of your implementation (programming langauge used, FM datastructure used, what was the input from a LLM if any, ...). Show you results clearly in tables and graphs. Do not forget to discuss your findings. The report should be in **PDF format !!!** The source code should be in SEPARATE, compressed archive file (program.tar.gz). **Do NOT include** your report.pdf file in this archive !

One group member submits the report and code zip file to Blackboard. The other group members submit a single page (pdf) with the name of the group members (including student number), and the name of the student that has submitted the report and code.

Deadline: March 31, 2025, 23:59 hrs