Implementation exercises for the course

Heuristic Optimization

Dr. Christian Camacho-Villalón and Dr. Thomas Stützle
Université Libre de Bruxelles — 2024

Implementation exercise sheet 2

Design and implement two stochastic local search algorithms for the linear ordering problem (LOP), building on top of the perturbative local search methods from the first implementation exercise. Apply your algorithms to the same instances as in the first implementation exercise.

The SLS algorithms can be based on either simple, hybrid, or population-based methods chosen among the ones described in the lectures. The two SLS methods chosen must belong to two different classes. For example, the first one could be a tabu search (a simple SLS method) and the second one could be an ACO algorithm (a population-based SLS method). To get inspiration for ways how to generate solutions, operators etc. you may consider algorithms that have been proposed in the literature. In the articles provided in the "reference" folder in this channel you will find a comprehensive review of heuristic algorithms for the LOP.

The two algorithms implemented should be evaluated on all instances of size 150 and be compared using statistical testing. In addition, on few instances the comparison should also be done by measuring run-time distributions. The implementation should make use of the iterative improvement algorithms from the first implementation exercise.

Please note that the experimental comparison should be among implementations that are coded in the same programming language, the algorithms should be compiled using the same compiler with same compiler flags, and the experiments should be run on the same computer under same conditions (e.g. no other compute or memory intensive programs are running on the same machine). In other words, the observed differences should be attributable to differences in the algorithms and not to differences in the experimental conditions.

The deadline for the implementation exercise is: **May 5, 2024 (23:59)**

Exercise 2.1

1. Run each algorithm once on each instance of size 150. Instances are available on TEAMS and are the same as used in the first implementation exercise. As termination criterion, for each instance use the average computation time it takes to run a full VND (implemented in the previous exercise) on the same instance size and then multiply this time by 100 (to allow for long enough runs of the SLS algorithms).

2. Compute for each of the two SLS algorithms and each instance the mean relative percentage deviation from the best known solutions.

3. Produce correlation plots of the relative percentage deviation for the two SLS algorithms (see lectures).

4. Determine, using statistical tests (in this case, the Wilcoxon test), whether there is a statistically significant difference between the relative percentage deviations reached by the two algorithms. Note: For applying the statistical test, the R statistics software can be used. The software can be download from `http://www.r-project.org/`.

5. [Optional] Measure, for each of the two implemented SLS algorithms on three instances qualified run-time distributions to reach sufficiently high quality solutions (e.g. the best-known solutions available or some solution value close to the best-known one such as 0.1%, 0.25%, or 1% worse than the best-known solution). Measure the run-time distributions across 25 repetitions using a cut-off time of 10 times the termination criterion above. As the instances take the first three instances of size 150.

6. Produce a written report on the implementation exercise:

   - Include a clear description of the implemented SLS algorithms, mentioning how they work, what are their algorithm components, and what were the computational results obtained by each of them. Justify also the choice of the parameter settings that are used and the choice of the method for generating (i) initial solutions and (ii) the iterative improvement algorithm.
   - Present the results as in the previous implementation exercise using tables and statistical tests.
   - Present the performance correlation plots and [optionally] present graphically the results of the analysis of the run-time distributions.

- Interpret appropriately the results (statistical tests, run-time distributions, correlation plots) and make conclusions on the relative performance of the algorithms across all the benchmark instances studied.

Recommended starting point for literature search:

The article below is the best starting point for exploring the available literature for examples of how the various existing SLS methods can be adapted to the LOP. We suggest to continue from this article the search for specific examples of SLS algorithms that have been applied to the LOP. A pdf of this article can be found in the Files tab of this channel on TEAMS.

- Rafael Martí, Gerhard Reinelt, and Abraham Duarte. *A benchmark library and a comparison of heuristic methods for the linear ordering problem*. Computational Optimization and Applications. 51(3):1297-1317 (2012).

Additional information:

- In order to pass the exam, you have to successfully complete this second implementation exercise.
- You need to do the implementation exercise by yourself — cooperation is forbidden.
- You have to submit the following items in **a zip folder with your name** via TEAMS:
  (i) a report in `pdf` format with scientific article structure (see examples provided on TEAMS) that concisely explains the implementation, reports the above mentioned tasks (averages, standard deviations, run-time distributions, correlation plots and results of statistical tests), and interprets concisely the observed results;
  (Note: some of the criteria to be evaluated in the report are: the use of a scientific article structure including abstract, intro, problem description, material and methods, results, conclusion; the use of tables and plots to present the obtained results; and the use of statistical tests to draw conclusion about the algorithms performance.)
  (ii) the source code of the implementation together with a `README` file explaining how to compile and/or execute the algorithms from a command line in Linux/MacOS;
  (Note: some of the criteria to be evaluated in code are: compilation/execution without errors, the use of structures, code efficiency, indentation and comments.)
  (iii) a simple `.txt` file with the raw data that was used for statistical testing.

  It is important to stress that your code should compile/run on Linux/MacOS without errors and produce the requested outputs when executed on the provided instances; otherwise the implementation exercise will be considered insufficient.
- As programming language, you may use preferably `C`, `C++`, or `Java`. While using `Python` for this exercise is also possible, we recommend against it because it is much more slower than the alternatives. The sample routines are available only in `C++`, but you are free to adapt them at your convenience. Please make sure that your code is properly commented and indented, and that the `README` file mentions the exact commands for its compilation and execution.

- **Happy coding :-D**