Implementation exercises for the course

Heuristic Optimization

F. Pagnozzi, Dr. T. Stützle
Université Libre de Bruxelles — 2021

Implementation exercise sheet 2

Implement two stochastic local search algorithms for the permutation flow-shop problem (PFSP) with the sum completion time objective (PFSP-CT) building on top of the constructive and 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 those 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 (simple SLS method) and the second one could be an ACO algorithm (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 for the same problem.

1. The most complete review of algorithms for the sum completion time PFSP including descriptions of ILS and IG algorithms.

   Q.-K. Pan, R. Ruiz. Local search methods for the flowshop scheduling problem with flowtime minimization. *European Journal of Operational Research*, 222(1): 31-43, 2012.

2. Examples of ACO algorithms.

   C. Rajendran and H. Ziegler. Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. *Computers & Industrial Engineering*, 48:789–797, 2005.

3. Examples of memetic algorithms.

   L. Y. Tseng and Y.-T. Lin. A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem. *International Journal of Production Economics*, 127:121–128, 2010.

   Y. Zhang, X. Li, and Q. Wang. Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. *European Journal of Operational Research*, 196(3):869-876, 2009.

4. A state-of-the-art algorithm for the PFSP-CT.

   Federico Pagnozzi and Thomas Stützle. Automatic design of hybrid stochastic local search algorithms for permutation flowshops problems. *European Journal of Operational Research*, 276(2):409–421, 2019.

The two algorithms implemented should be evaluated on all instances provided for the first implementation exercise and be compared using statistical tests. In addition, on few instances run-time distributions can be measured and be used for comparing the two algorithms. The implementation should make use of appropriately selected iterative improvement algorithms from the first implementation exercise. Justify the choice of the iterative improvement algorithm you use from the ones you implemented for the first implementation exercise.

Please note that the experimental comparison should be run under same conditions (programming language, compiler and compiler flags, similar load on computer etc.) for the two algorithms. In other words, the observed differences should be attributable to differences in the algorithms and not to differences in the experimental conditions.

Exercise 2.1 Implementation, deadline May 13, 2021

1. Run each algorithm ten times on each instance. 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 first exercise) on instances of the same size (50, 100, and 200) and then multiply this time by 500 (to allow for long enough runs of the SLS algorithms).

2. Compute for each of the two SLS algorithms and each instance the average percentage deviation from the best known solutions. For each instance size, also compute the average percentage deviation across all instances.

3. Produce correlation plots of the average relative percentage deviation for the two SLS algorithms (see lectures), separating between the instance sizes (either two separate plots or clearly marking in colors, separating the different instances size, in one plot).

4. Determine, using statistical tests (in this case, the Wilcoxon test), whether there is a statistically significant difference between the mean relative percentage deviations reached by the two algorithms for each instance size. 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 the first 2 instances of size 50 and size 100 jobs, 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 $\{0.1, 0.5, 1, 2\}\%$ above the best-known solutions). Measure the run-time distributions across 25 repetitions using a cut-off time of 10 times the termination criterion above.

6. You have to submit the following items in **a zip folder with your name** via TEAMS:

   - 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, and results of statistical tests), interprets concisely the observed results, and answers the questions posed above;
     (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 to present the obtained results; and the use of statistical tests to draw conclusion about the algorithms performance.)
   - 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 the code are: compilation/execution without errors, the use of structures, code efficiency, indentation and comments.)
   - 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.
   - 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 slower than the alternatives. 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.

Deadline for the implementation exercises:

- **May 13, 2025**