Implementation exercises for the course

Heuristic Optimization

F. Pagnozzi, Dr. T. Stützle
Université Libre de Bruxelles — 2017

Implementation exercise sheet 2

Implement two stochastic local search algorithms for the permutation flow-shop problem (PFSP) with the sum weighted completion time objective (PFSP-WCT) 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 or similar problems. In fact, the PFSP-SWC has not been very frequently treated in the literature. For inspiration you may also check papers on the related PFSP with sum completion time (or flowtime) objective. Some references are given below.

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.

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 should 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 17, 2017

1. Run each algorithm five times on each instance. Instances are available from `http://iridia.ulb.ac.be/~stuetzle/Teaching/HO/` 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 or 100) 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 differences 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. Measure, for each of the two implemented SLS algorithms on the first 5 instances of size 50 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. Produce a written report on the implementation exercise:

   - Please make sure that for each implemented SLS algorithm it is clearly described how it works and which are the used algorithmic components and that the computational results are carefully interpreted. Justify also the choice of the parameter settings that are used and the choice of the method for generating initial solutions and the iterative improvement algorithm for the SLS algorithms.
   - Present the results as in the previous implementation exercise using tables and statistical tests.
   - Present graphically the results of the analysis of the run-time distributions and the performance correlation plots.
   - 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.

Additional information on the implementation exercise:

- Recall that the completion of the implementation task is a pre-condition for the examination.
- Every student sends (i) the above mentioned report in pdf format and (ii) the source code of the implementation to `federico.pagnozzi@ulb.ac.be`. Please send one single archive (zip or tar.gz) with all implementations. A `README` file should explain how to compile and run the code.
- As programming language, you may use `C`, `C++`, or `Java`.
- Please take care that the code is reasonably documented and mention the exact commands for compilation in a `README` file.
- For each of the implemented SLS algorithms, provide a concise description of at least one page of the main features of the algorithm (e.g. choice of initial solution, iterative improvement algorithm, crossover operator in case of MA, perturbation operators, etc.)
- The articles mentioned above are available from the lecture's webpage.

Deadline for the implementation exercises:

- **May 17, 2017**