Université libre de Bruxelles
École Polytechnique de Bruxelles
M-IRIFS

INFO-H413 - Heuristics optimization - 2023/2024

Linear ordering problem

Implementation exercise 2

Student:
Younes El Mokhtari

Teacher:
Thomas STÜTZLE

May 9, 2024

Contents

Abstract

In this study, we report the results of stochastic local search algorithms in C++ to solve the linear ordering problem (LOP) for various instances. We used iterated local search algorithm with random initial solution, insert based local search (ILS) and exchange based perturbation. We also employed memetic algorithm (MA) with random initial population of fixed size, insert based local search, CX (cycle crossover) recombination, exchange mutation with fixed mutation rate and rank selection. To evaluate the performance of each algorithm across instances of size 150, statistical analysis is performed using metrics such as average percentage deviation from the best-known solutions and total computation time. Statistical tests such as the Student t-test and Wilcoxon test are also employed to assess wheter there is statistically significant difference between the solutions generated by each algorithm.

Keywords: linear ordering problem, stochastic local search, iterated local search algorithm, memetic algorithm, Student t-test, Wilcoxon test, C++.

1 Introduction

The goal of this second implementation exercise was to design and implement two stochastic local search (SLS) algorithm from two different classes (simple, hybrid or genetic) in order to solve the linear ordering problem. In this study, we choose to implement ILS (hybrid) and MA (genetic). A set of instances with sizes 150 and 250, accompanied by corresponding best-known solutions were given for this exercise and served as test cases for the implemented algorithms. The termination criterion for the algorithms was based on the average computation time it takes to run a full VND implemented in the first exercise, on the same instance and multiplied by 100 in order to allow for long enough runs of SLS algorithms. To evaluate the performance of each algorithm, we also computed statistics such as average percentage deviation and computation time for all instances and employed statistical analysis using Student-t test and Wilcoxon test to determine if there were significant differences between the solutions. We produced also correlation plots between two algorithms and for all instances computed.

The paper is organized as follows: Section 2 describe the linear ordering problem, Section 3 explain the implemented algorithms, Section 4 interpret the results obtained and Section 5 is the conclusion.

1.1 Structure

The file tree below contains a short description of the implementation directories:

```
linear-ordering-problem
├── best_known - best-known scores for each instance
├── doc - report
├── instances - instances of size 150 and 250
├── max_runtime - termination criterion for sls algorithms
├── references - articles
├── scripts
│   ├── correlation_plots.r
│   ├── ie.py
│   └── statistical_tests.r
├── src
│   ├── lop
│   └── sls
├── statistics
│   ├── ie1
│   └── ie2
├── lop
└── Makefile
```

2 Problem description

2.1 Linear ordering problem (LOP)

The Linear Ordering Problem (LOP) is a combinatorial optimization challenge focused on arranging a set of items in a specific linear order to optimize a given objective function. In this problem, the task is to find a permutation of elements from a given set such that certain criteria are optimized.

Mathematically, the LOP can be defined as follows: Given an $n \times n$ matrix C, the objective is to find a permutation π of indices from 1 to n that maximizes the sum of elements in the upper triangle of C. This objective function is represented as:

$$f(\pi) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} c_{\pi_i \pi_j} \tag{1}$$

where $c_{\pi_i \pi_j}$ denotes the element of the matrix C located at row π_i and column π_j after the permutation π has been applied. [1]

The Linear Ordering Problem has significant applications across various disciplines. In economics, it assists in the triangularization of input-output matrices, facilitating efficient resource allocation. In sociology, it aids in understanding social hierarchies and preferences. Moreover, in graph theory, linear orderings are crucial for representing partial orders and scheduling tasks. Additionally, in archaeology, the problem finds utility in establishing chronological sequences, as evidenced by the application of the Harris Matrix. [2]

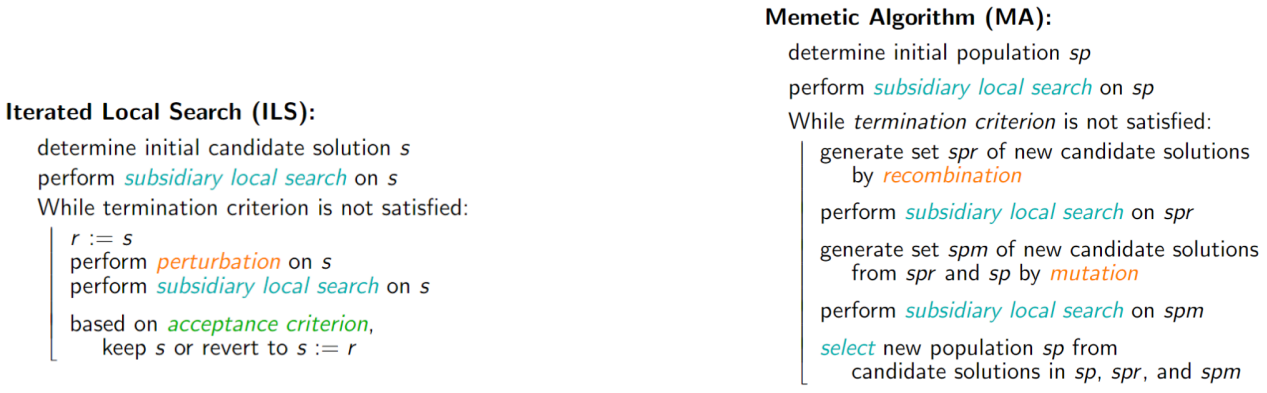
3 Material and methods

3.1 Data

The data used in this study consists of instances provided by the teacher. These instances vary in size between 150 and 250. For this second implementation exercise, only instances of size 150 are used. These can be found in the folder `./instances/`.

3.2 SLS algorithms

For this second implementation exercise, we implemented two stochastic local search (SLS) algorithm from two different classes (simple, hybrid or genetic) in order to solve the linear ordering problem. We selected ILS (hybrid) and MA (genetic) as they gave the best results shown in the reference [1]:



(a) Iterated local search

(b) Memetic algorithm

Figure 1: SLS algorithms

3.2.1 Iterated local search (ILS)

The implemented Iterated Local Search (ILS) algorithm is a metaheuristic optimization technique that iteratively improves a candidate solution through a combination of local search and perturbation phases. The key components of our ILS algorithm are as follows:

- **Initialization**: The algorithm starts by generating a random initial solution with a seed calculated as the sum of elements in the instances. This seed-based initialization ensures the creation of diverse initial solutions and maintains consistency across multiple runs.
- **Local Search**: The local search procedure employs a first improvement insert-based strategy to explore the largest neighborhood possible where we return the best solution after visiting all j elements for a chosen i.
- **Perturbation**: To introduce diversification, we opted for a random exchange strategy where we explore randomly the neighbourhood. We choose this pertubation strategy since it is a move that cannot be reversed with insert movements in a single step, as mentioned in the reference [1], and also because it is complementary to our local search. The number of exchange moves used in a perturbation is a parameter in our algorithm. We choose the first improved solution to ensure that our pertubated solution is always improving.
- **Acceptance Criterion**: The acceptance criterion evaluates the perturbed solution based on its objective function value. If the perturbed solution improves upon the current solution, it is accepted for further exploration; otherwise, it rolls back to the last best solution.
- **Termination**: The algorithm terminates if the elapsed time is superior or equal to the max runtime computed in Section 4.1.1.

3.2.2 Memetic algorithm (MA)

The implemented Memetic Algorithm (MA) is a hybrid optimization technique that combines evolutionary algorithms with local search methods to efficiently explore the search space. The components and justifications for the MA algorithm are as follows:

- **Initialization**: Similar to ILS, the MA algorithm initializes the population with random solutions using the same seed-based approach to ensure consistency and randomness.
- **Subsidiary Local Search**: The local search integrated into the MA algorithm follows the same first improvement insert-based strategy as in ILS to explore the largest neighbourhood.
- **Recombination**: The recombination operator in the MA algorithm employs cycle crossover (CX), as it has been shown to produce the best results based on the reference [1].
- **Mutation**: The mutation operator utilizes an first improvement exchange strategy with a mutation rate of 0.3. This rate is chosen empirically. The exchange strategy was chosen as it is complementary to the subsidiary local search.
- **Selection**: The selection operator in the MA algorithm is rank-based, selecting the best solutions from the population. A population size of 25 was set empirically.
- **Termination**: Similar to ILS, the MA algorithm terminates if the elapsed time is superior or equal to the max runtime computed in Section 4.1.1.

The parameter settings and operator choices in both algorithms are guided by empirical experimentation, previous research findings, and problem characteristics to ensure effective performance and convergence to high-quality solutions.

4 Results

All these results can be obtained using the python script (`./scripts/ie.py`) or by running `make ie2` with Makefile. The results are accessible in the `./statistics/ie2/` folder.

4.1 Statistics

4.1.1 Termination criterion

The termination criterion for the SLS algorithms was based on the average computation time it takes to run a full VND implemented in the first exercise, on the same instance and multiplied by 100 in order to allow for long enough runs of SLS algorithms. For all instances of size 150, we have in average 400 seconds of runtime. These run times can be found in `./max_runtime/max_runtime.txt`.

4.1.2 Average percentage deviations and total computation time

For each algorithm, the average deviation (in %) and total computation time (in second) is calculated on instances of size 150. The Table 1 summarise these informations for ILS and MA. Both ILS and MA give similar results:

Configuration	**Average deviation**	**Total computation time**
ils_random_insert_random	2.211367	13799.884000
ma_random_insert_cx_exchange_rank	1.161147	13799.884000

Table 1: Average deviation (in %) and total computation time (in second) for each configuration

4.2 Statistical tests

These statistical tests use a statement called the null hypothesis which in this case is: "the median of the differences between results in two experiments is zero". We fix the significance level α to 0.05. If the computed p-value is below α, the null hypothesis is rejected which in this case means that there is a significant difference between experiments. An R script (`./statistical_tests.r`) was written in order to apply these tests to all possible pairs of experiments. The output was saved in the `./statistics/ie2/statistical_tests/`.

4.2.1 Student-t test and Wilcoxon test

The results of the statistical tests shown in Table 2 indicate significant differences between the ILS and MA algorithms. The Student-t test yielded an extremely small p-value (7.53×10^{-33}), providing strong evidence against the null hypothesis and suggesting a significant difference in performance between the two algorithms. Similarly, the Wilcoxon test also produced a small p-value (1.71×10^{-14}), further supporting the rejection of the null hypothesis and indicating a statistically significant difference in performance between the ILS and MA algorithms.

Algorithm	ILS	MA
ILS	0	7.53×10^{-33}
MA	0	0

(a) Student-t

Algorithm	ILS	MA
ILS	0	1.71×10^{-14}
MA	0	0

(b) Wilcoxon

Table 2: Statistical tests

4.3 Correlation plot

To assess the correlation between the Iterated Local Search (ILS) and Memetic Algorithm (MA) performance, we created a scatter plot illustrating the relationship between their relative deviations. Additionally, we calculated the correlation coefficient, which yielded approximately 0.557. This positive correlation coefficient indicates a moderate positive linear relationship between the relative deviations for ILS and MA. In other words, as the relative deviation for ILS tends to increase or decrease, the relative deviation for MA also tends to increase or decrease, although not perfectly.

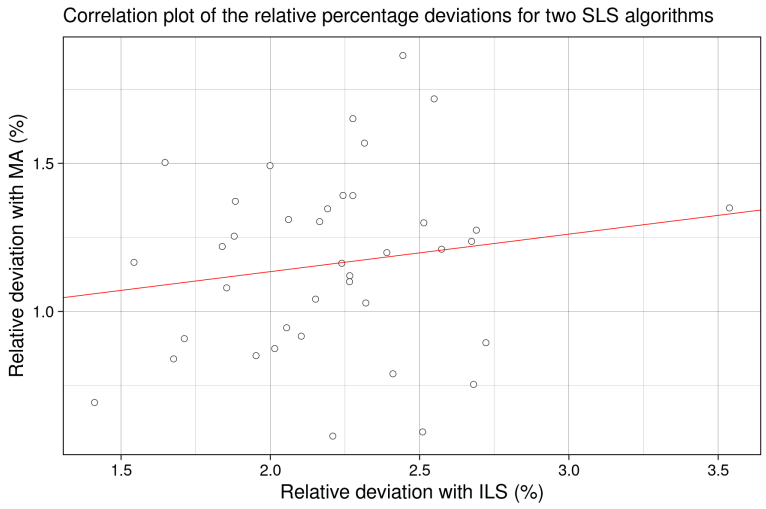


Figure 2: Correlation plot

5 Conclusion

A computational comparison was conducted on 38 instances of size 150 to evaluate the performance of two SLS algorithms, namely Iterated Local Search (ILS) and Memetic Algorithm (MA), for solving the LOP. The best results were achieved by the MA, followed closely by ILS.

References

[1] *The Linear Ordering Problem: Instances, Search Space Analysis and Algorithms*, Tommaso Schiavinotto & Thomas Stützle

[2] *Stochastic Local Search Foundations and Applications*, Elsevier, 2004, Holger H. Hoos & Thomas Stützle