Université libre de Bruxelles
École Polytechnique de Bruxelles
M-IRIFS

INFO-H413 - Heuristics optimization - 2023/2024

Linear ordering problem

Implementation exercise 1

Student:
Younes El Mokhtari

Teacher:
Thomas STÜTZLE

April 7, 2024

Contents

Abstract

In this study, we report the results of using iterative improvement algorithms and variable neighbourhood descent algorithms in C++ to solve the linear ordering problem (LOP) for various instances. We used iterative improvement algorithms with various pivoting rules (first and best improvement), neighbourhoods (transpose, exchange, and insert) and initial solutions (random permutation and the Chenery-Watanabe heuristic). We also employed a two-variable neighbourhood descent method, with a first improvement and two neighborhood orders (transpose, exchange, then insert and transpose, insert, then exchange). To evaluate the performance of each algorithm combination across all instances, 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 combination.

Keywords: linear ordering problem, iterative improvement, variable neighbourhood descent, Chenery-Watanabe heuristic, Student t-test, Wilcoxon test, C++.

1 Introduction

The goal of this first implementation exercise was to implement iterative improvement algorithms (Exercise 1.1) and variable neighborhood descent algorithms (Exercise 1.2) with different rules (initial solutions, neighbourhood and pivoting) to solve the linear ordering problem. 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.To evaluate the performance of each algorithm, we 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.

The paper is organized as follows: Section 2 describe the linear ordering problem, Section 3 explain the data used the 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
├── references - articles
├── src
│   ├── rules - rules like inital solutions, neighbourhood and pivoting
│   ├── configuration.cpp
│   ├── instance.cpp
│   └── main.cpp
├── statistics
│   ├── experiments - relative deviations for each algorithm for each instance
│   ├── reports - average deviation for each algorithm for all instances
│   └── statistical_tests - Student-t test and Wilcoxon test
├── lop
├── Makefile
├── statistical_tests.r
└── statistics.sh
```

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. These can be found in the folder ./instances/.

3.2 Algorithms

For the implementation of algorithms, various combinations were considered:

- Iterative Improvement (II) algorithms:
  - Initial solutions: random permutation and Chenery-Watanabe heuristic
  - Neighborhoods: transpose, exchange, and insert
  - First improvement and best improvement pivoting rules
- Variable Neighborhood Descent (VND) algorithms:
  - Chenery-Watanabe heuristic
  - Neighborhood orderings: transpose-exchange-insert and transpose-insert-exchange
  - Applied only to first improvement iterative algorithms

This results in 12 algorithms for iterative improvement and 2 for variable neighbourhood descent algorithms so 14 in total. The implementations of each rule can be found in ./src/rules/ and each algorithm in ./src/algorithm.cpp.

4 Results

The results from exercise 1.1 and exercise 1.2 are merged for convenience. A bash script (./statistics.sh) was used to save the differents results in the folder ./statistics/.

4.1 Statistics

For each combination algorithm, the average deviation (in %) and total computation time (in second) is calculated on all instances. The Table 1 summarise all the 12 combinations of II algorithms for exercise 1.1 and also all the 2 combinations VND algorithms for exercise 1.2.

4.1.1 Average percentage deviations and total computation time

The transpose neighbourhood has the best computation times, but the worst solutions (highest average percentage deviation). The exchange neighbourhood provides the best solutions but with high computation time.

Configuration	**Average deviation**	**Total computation time**
ii_cw_exchange_best	4.394455	3247.509990
ii_cw_exchange_first	30.963442	1.370133
ii_cw_insert_best	13.742654	260.923158
ii_cw_insert_first	30.985910	1.294789
ii_cw_transpose_best	30.109363	15.094090
ii_cw_transpose_first	30.977256	1.273179
ii_random_exchange_best	3.624083	3852.671460
ii_random_exchange_first	35.407036	0.053327
ii_random_insert_best	14.067337	257.784736
ii_random_insert_first	35.416426	0.074736
ii_random_transpose_best	34.225995	19.884247
ii_random_transpose_first	35.417441	0.051101
vnd_cw_transpose-exchange-insert_first	30.924140	1.414506
vnd_cw_transpose-insert-exchange_first	30.892497	1.418386

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/statistical_tests/.

Since there is no space to display the tables, here is the mapping for rows and columns

- V1 → `ii_cw_exchange_best`
- V2 → `ii_cw_exchange_first`
- V3 → `ii_cw_insert_best`
- V4 → `ii_cw_insert_first`
- V5 → `ii_cw_transpose_best`

- V6 → `ii_cw_transpose_first`
- V7 → `ii_random_exchange_best`
- V8 → `ii_random_exchange_first`
- V9 → `ii_random_insert_best`
- V10 → `ii_random_insert_first`
- V11 → `ii_random_transpose_best`
- V12 → `ii_random_transpose_first`
- V13 → `vnd_cw_transpose-exchange-insert_first`
- V14 → `vnd_cw_transpose-insert-exchange_first`

4.2.1 Student-t test and Wilcoxon test

The random and Chenery-Watanabe initial solutions have statically different qualities for all combinations. The random initial solution is preferable because it gives a lesser deviation compared to Chenery-Watanabe (V3-V9).

All neighborhoods have statically different qualities. The table 1 shows that the exchange neighborhood give the better solutions. It is weird because insert neighborhood should give better solutions since it has a larger neighbourhood. Last is transpose giving the less good results.

The first and best pivoting rules have statically different qualities. Best is giving always better results except for transpose.

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
1	0	5.69e-72	1.3e-73	6.8e-72	8.75e-71	4.94e-72	8.63e-11	1.8e-67	7.4e-68	1.74e-67	1.76e-66	1.73e-67	5.21e-72	5.3e-72
2	0	0	8.79e-59	0.0246	9.33e-32	0.0464	8.52e-73	8.4e-15	3.38e-59	7.65e-15	5.29e-10	7.59e-15	0.000122	0.000148
3	0	0	0	1.05e-58	4.65e-57	7.53e-59	1.57e-74	7.08e-54	0.0186	6.9e-54	1.71e-52	6.84e-54	7.82e-59	6.94e-59
4	0	0	0	0	9.15e-31	0.46	1.01e-72	1.15e-14	3.58e-59	1.05e-14	7.21e-10	1.04e-14	0.000144	4.59e-06
5	0	0	0	0	0	8.44e-33	1.24e-71	1.62e-18	2.22e-57	1.48e-18	7.31e-14	1.47e-18	1.86e-31	1.07e-27
6	0	0	0	0	0	0	7.73e-73	1e-14	3.13e-59	9.12e-15	6.24e-10	9.05e-15	5.56e-06	6.34e-07
7	0	0	0	0	0	0	0	3.93e-68	6.64e-72	3.81e-68	4.36e-67	3.77e-68	7.97e-73	8.26e-73
8	0	0	0	0	0	0	0	0	1.25e-53	0.0128	5.16e-32	0.00898	5.17e-15	5.51e-15
9	0	0	0	0	0	0	0	0	0	1.21e-53	4.6e-52	1.2e-53	3.1e-59	3.75e-59
10	0	0	0	0	0	0	0	0	0	0	6.7e-32	0.123	4.7e-15	5.02e-15
11	0	0	0	0	0	0	0	0	0	0	0	5.88e-32	3.31e-10	3.23e-10
12	0	0	0	0	0	0	0	0	0	0	0	0	4.67e-15	4.98e-15
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0.05
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2: Student-t statistical test

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14
1	0	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14	2.14e-09	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14
2	0	0	1.72e-14	0.0389	1.72e-14	0.248	1.72e-14	4.16e-14	1.72e-14	4.16e-14	3.7e-12	4.16e-14	9.6e-14	1.08e-09
3	0	0	0	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14	0.0571	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14
4	0	0	0	0	1.72e-14	0.117	1.72e-14	4.33e-14	1.72e-14	4.33e-14	5.08e-12	4.33e-14	1.85e-14	1.71e-14
5	0	0	0	0	0	1.72e-14	1.72e-14	1.78e-14	1.72e-14	1.78e-14	6.58e-14	1.78e-14	1.72e-14	2e-14
6	0	0	0	0	0	0	1.72e-14	4.49e-14	1.72e-14	4.49e-14	4.26e-12	4.49e-14	1.72e-14	1.71e-14
7	0	0	0	0	0	0	0	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14
8	0	0	0	0	0	0	0	0	1.72e-14	0.0638	1.72e-14	0.409	3.86e-14	3.57e-14
9	0	0	0	0	0	0	0	0	0	1.72e-14	1.72e-14	1.72e-14	1.72e-14	1.72e-14
10	0	0	0	0	0	0	0	0	0	0	1.72e-14	0.00334	3.86e-14	3.57e-14
11	0	0	0	0	0	0	0	0	0	0	0	1.72e-14	2.42e-12	2.1e-12
12	0	0	0	0	0	0	0	0	0	0	0	0	3.86e-14	3.57e-14
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0.000594
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 3: Wilcoxon statistical test

5 Conclusion

For each combinaison of algorithm, we implementede successfully and obtained results for each algorithm.

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