Université Libre de Bruxelles

ECOLE POLYTECHNIQUE DE BRUXELLES

ULB

Heuristic optimisation INFO-H413

Implementation Exercise Report Linear Ordering Problem (LOP)

Student :
Achten Alexandre 494484

Professors :
Stützle Thomas
Camacho-Villalón Christian

May 2024

Academic year 2023-2024

Abstract

"The LOP is $\mathcal{NP}$-hard, that is, we cannot expect to find a polynomial time algorithm for its solution" [SS04].
In this report, different algorithms will be exposed. Their mode of operation will be explained and the associated results (Section 4) will be presented. A statistical analyse (Subsection 4.3) between these results will be done using the Paired t-test and the Wilcoxon signed-rank test.
For the classical methods (Best and First) the initial solution doesn't have a signif- icant impact on the most of the algorithms. Concerning the VND's algorithms, the order of evaluation of the neighbourhoods does have an impact on the execution time. The best-time iterative improvement algorithm is the transpose pivoting rule algo- rithm, but as it's relative deviation is really high, the First exchange CW is the fastest one with decent results. The most accurate iterative improvement algorithm, that have the lowest average relative deviation, is the First insert CW. For the stochastic local search algorithms, the MA and ILS present both good re- sults and we can't find a significant difference between them. But it is important to mention that the ILS find faster better solutions (as shown in Figure 8).

Contents

1 Introduction

The linear ordering problem (LOP) is a classical problem that arises in several different fields (economics, archeology,...). Given a matrix C nxn and a permutation π of the columns and indices $\{1, ..., n\}$, the sum of the elements in the upper right triangle must be maximised. More formally, this formula must be maximised:

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

"The LOP is $\mathcal{NP}$-hard, that is, we cannot expect to find a polynomial time algorithm for its solution" [SS04].
In this report, different algorithms will be exposed. Their mode of operation will be explained and the associated results (Section 4) will be presented. A statistical analyse (Subsection 4.3) between these results will be done using the Paired t-test and the Wilcoxon signed-rank test.

2 Iterative Improvement methods

"The $\mathcal{N}_X(\pi)$ (the neighborhood of a permutation π), is defined to be the set of all permutations that can be obtained by applying one interchange move to π" [SS04]. Where the interchange moves are the pivoting rules we are discussing in Subsection 2.3.
Iterative improvement methods mean that the algorithm tries to find a better solution within the neighbourhood, and from the new solution, searches through the neighbourhood again.
The general operation of the iterative improvement method is explained here :

π := GenerateInitialSolution()
if π is not a local optimum do
    pick a neighbour π' $\in \mathcal{N}$ such that $f(\pi') > f(\pi)$
    $\pi := \pi$'

Where $f(\pi)$ is the objective function to maximise.
The two main algorithms used here are first-improvement and best-improvement. Each has its advantages and disadvantages.

2.1 First-improvement

The First-Improvement method is an iterative improvement method.
This method is based on the principle that when evaluating neighbours, whenever it finds a better neighbour, it chooses that solution without visiting all the other possibilities. And it will continue to evaluate neighbours from this new solution until no more better neighbours are found. It's initial solution can be generated randomly or using the Chenery and Watanabe (CW) heuristic construction method (Subsection 2.4). The permutation can be modified in several ways, in this report 3 pivoting rules are used and are presented in the Subsection 2.3. Each pivoting rule has it's own neighbourhood.

2.2 Best-improvement

This method is based on the principle of exhaustive search of all neighbours. It remembers the best neighbour of all and returns it when it has finished evaluating each neighbour. It's initial solution can be generated randomly or using the Chenery and Watanabe (CW) heuristic construction method (Subsection 2.4). The permutation can be modified in several ways, in this report 3 pivoting rules are used and are presented in the Subsection 2.3. Each pivoting rule has it's own neighbourhood.

2.3 Pivoting rules

Each pivoting rule have it's own neighbourhood, so it's a very important choice to choose the kind of pivoting rule. These pivoting rules can be applied on any vector.

2.3.1 Exchange

The exchange pivoting rule consist in swapping any element of the vector with any other element. Let i be the elements to swap, the neighbourhood $\mathcal{N}_i$ are all $j \neq i$ elements such that $\forall j \in \pi$. An example is given in the Figure 1.

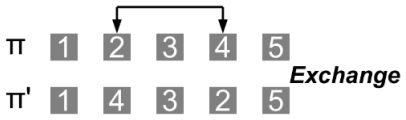

Figure 1: Exchange example

2.3.2 Transpose

The transpose pivoting rule consist in swapping an element with one of it's neighbours. Let i be the element to transpose, the neighbourhood $\mathcal{N}_i$ are all $j \neq i$ elements such that $\forall j \in \{i-1, i+1\}$. An example is given in the Figure 2.

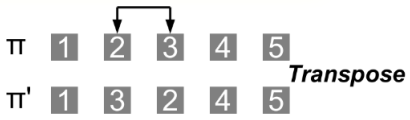

Figure 2: Transpose example

2.3.3 Insert

The insert pivoting rule insert the first element at the place of the second one, shifting the entire vector. An example is given in the Figure 3.

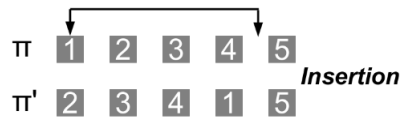

Figure 3: Insert example

2.4 Initial solution

At the beginning of any algorithm that searches for a solution, an initial solution is needed.

2.4.1 Random Solution

The random initial solution consists in creating a random solution that's acceptable. Concretely it generates a random vector that represents the permutation π.

2.4.2 Chenery and Watanabe

The Chenery and Watanabe is a constructive heuristic that creates the initial solution based on the attractiveness of each row, placing the most attractive one first. The attractiveness a_i of the row i is defined as the sum of the costs of each element in this row.

$$a_i = \sum_{j=1}^{n} c_{ij} \tag{2}$$

[MRD09]
Where c_{ij} is the cost of the element at row i and column j.

2.5 Variable Neighbourhood Descent

The Variable Neighbourhood Descent (VND) method is obtained when the change between neighbours is performed in a deterministic way, during the local search [MR11].
It follows the following schema :

k neighbourhoods $\mathcal{N}_1, ..., \mathcal{N}_k$
π := GenerateInitialSolution()
i:=1
repeat
    choose the first improving neighbour π' $\in \mathcal{N}_i(\pi)$
    if not $\exists\pi$' then i:=i+1
    else $\pi := \pi$' and i:=1
until $i > k$

In other words, if no other better neighbour is found, the algorithm searches through another neighbourhood by changing the pivoting rule.

In this report, two different descent are used :

- VND1 : Transpose → Exchange → Insert
- VND2 : Transpose → Insert → Exchange

In addition of these two descent, the two algorithms (VND1 and VND2) use the first improvement method and the Chenery and Watanabe initial solution.

3 Stochastic Local Search methods

Stochastic Local Search (SLS) methods are used to effectively escape from local optima of a given function. They use probabilistic improvements and sometimes accept small worsening of the objective function. They use iterative improvements methods as local search algorithm.

3.1 Iterated Local Search

ILS iterates in a particular way over the local search process by applying three main steps: (i) perturb a locally optimal solution, (ii) locally optimize it with the local search chosen and (iii) choose, based on some acceptance criterion, the solution that undergoes the next perturbation phase. [SS04]
It follows the following schema :

$\pi \leftarrow GenerateInitialSolution()$;
$\pi \leftarrow LocalSearch(\pi)$;
repeat
    $\pi' \leftarrow Perturbation(\pi)$;
    $\pi' \leftarrow LocalSearch(\pi')$;
    $\pi \leftarrow AcceptanceCriterion(\pi, \pi', history)$;
until termination condition met;

The termination condition is the elapsed time and it is set at 100 times the worst case in the VND algorithm. In this case 250 seconds. The LocalSearch function uses the VND2 algorithm. The perturbation is done by exchanging randomly some elements of the permutation π. In this algorithm it is set to 8 perturbations. The acceptance criterion is the following : a new local optimum is accepted if the objective function $f(\pi') > (1 - \varepsilon)f(\pi)$. The ε value is set to $\varepsilon = 0.0001$, which allows small worsening of the objective function in order to get out of local optima's.[SS04]

3.2 Memetic algorithm

MAs are evolutionary algorithms that are intimately coupled with local search algorithms, resulting in a population-based algorithm that effectively searches in the space of local optima. [SS04] It's basically a genetic algorithm coupled with a local search procedure.
It follows the following algorithm :

Population $\leftarrow$\{\};
for $i = 1...m$ do \{m is the number of individuals\}
    $\pi \leftarrow LocalSearch(GenerateRandomSolution())$;
    Population $\leftarrow$ Population $\cup\{\pi\}$;
end for
repeat
    Offspring $\leftarrow$ \{\};
    for $i \leftarrow 1...\#crossovers$ do
        draw π^a, π^b from Population
        Offspring $\leftarrow$ Offspring $\cup\{LocalSearch(Crossover(\pi^a, \pi^b))\}$;

end for
for $i \leftarrow 1...\#mutations$ do
draw π^a from Population
Offspring $\leftarrow$ Offspring $\cup\{LocalSearch(Mutate(\pi^a))\}$;
end for
Population $\leftarrow$ SelectBest(Population $\cup$ Offspring, m);
until termination condition met;

Where termination condition is the elapsed time and it is set at 100 times the worst case in the VND algorithm. In this case 250 seconds. The local search uses the VND2 algorithm. The individuals to which crossover and mutation are applied are chosen randomly according to a uniform distribution.The crossover operator takes two individuals of the current population and combines them into a new individual, while the mutation operator introduces a perturbation into an individual.[SS04]
A population of 30 individuals is used, for the initial generation, they are all randomly generated. The CX crossover is used, it consist in keeping elements in common to the both parents. For the other elements, the CX operator chooses randomly an empty position i and a parent π^1, determine an element $a = \pi^1_i$ that in turn is assigned to the offspring in position i. In the second parent, π^2, a different element $b = \pi^2_i$ occupies this same position i. The element b is then copied to the offspring in the position occupied by b in π^1. This process iterates until all the position are filled. An example of the crossover is presented at Figure 4.[SS04]
The mutations are exchange operations that are repeated 5 times on a random individual of the population.

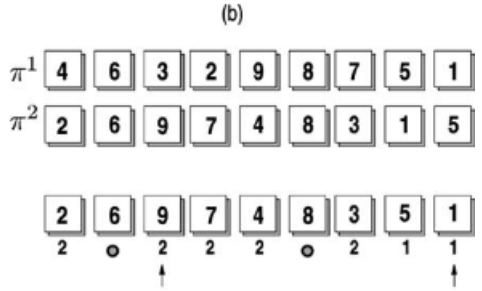

Figure 4: CX crossover, the positions marked with a circle are common to the parents [SS04]

4 Results

All the results are obtained on the XLOLIB dataset that contains 78 instances of the problem. There are 39 instances with $n = 150$ and 39 with $n = 250$. The relative deviation compared to the best known solution is denoted Δ and the time to finish the algorithm is t. The tables below show the average relative deviation Δ_{avg} and the average time t_{avg}.

Computationnal environnement :

- CPU : Intel i7-1065G7 (8 cores) 1.497GHz

- OS : Ubuntu 22.04.4 LTS on Windows 10 x86_64
- Kernel : 5.15.146.1-microsoft-standard-WSL2
- RAM : 7822MB
- Compiler : gcc with -03 flag

4.1 Iterative improvement

First improvement
The first improvement algorithms were tested on the $n = 150$ and the $n = 250$ instances. The results are presented in the Table 1 and Table 2.
The names of the algorithms use this format :
"name_of_improvement" "name_of_pivoting_rule" "initial_solution".

Algorithm	Δ_{avg} (%)	t_{avg} (s)
First exchange random	3.783	0.672
First insert random	2.090	1.520
First transpose random	35.992	0.008
First exchange CW	3.480	0.638
First insert CW	1.989	1.413
First transpose CW	20.020	0.006

Table 1: First improvement algorithms on instances of $n = 150$

Algorithm	Δ_{avg} (%)	t_{avg} (s)
First exchange random	3.591	5.914
First insert random	2.228	14.633
First transpose random	33.220	0.033
First exchange CW	3.271	5.993
First insert CW	1.936	14.346
First transpose CW	18.667	0.035

Table 2: First improvement algorithms on instances of $n = 250$

Best improvement
For the best improvement algorithms, instances of both sizes ($n = 150$ and $n = 250$) worked in reasonable time. The results for $n = 150$ are presented in Table 3 and for $n = 250$ at Table 4.

Algorithm	Δ_{avg} (%)	t_{avg} (s)
Best exchange random	3.635	11.755
Best insert random	2.187	24.757
Best transpose random	35.864	0.097
Best exchange CW	3.521	10.825
Best insert CW	2.037	22.186
Best transpose CW	19.937	0.107

Table 3: Best improvement algorithms on instances of $n = 150$

Algorithm	Δ_{avg} (%)	t_{avg} (s)
Best exchange random	3.463	157.980
Best insert random	2.370	320.268
Best transpose random	34.851	0.638
Best exchange CW	3.272	150.886
Best insert CW	2.106	314.345
Best transpose CW	18.526	0.611

Table 4: Best improvement algorithms on instances of $n = 250$

VND
The VND algorithm uses CW initial solutions. The results are shown in the Table 5 and Table 6.

Algorithm	Δ_{avg} (%)	t_{avg} (s)
VND1	1.970	2.397
VND2	2.001	1.495

Table 5: VND algorithms on instances of $n = 150$

Algorithm	Δ_{avg} (%)	t_{avg} (s)
VND1	2.030	23.214
VND2	2.027	13.770

Table 6: VND algorithms on instances of $n = 250$

4.2 Stochastic local search

The methods presented in this section are applied on $n = 150$ instances.

Memetic Algorithm (MA)
The average number of generations generated at the end of the algorithm is denoted by Gen_{avg}. The results on instances of $n = 150$ are presented in the Table 7.

Algorithm	Δ_{avg} (%)	t_{avg} (s)	Gen_{avg}
Memetic	0.619	253.817	23.590

Table 7: Memetic algorithm on instances of $n = 150$

The solution quality distribution (SQD) for 3 different runtime is presented at the Figure 5.

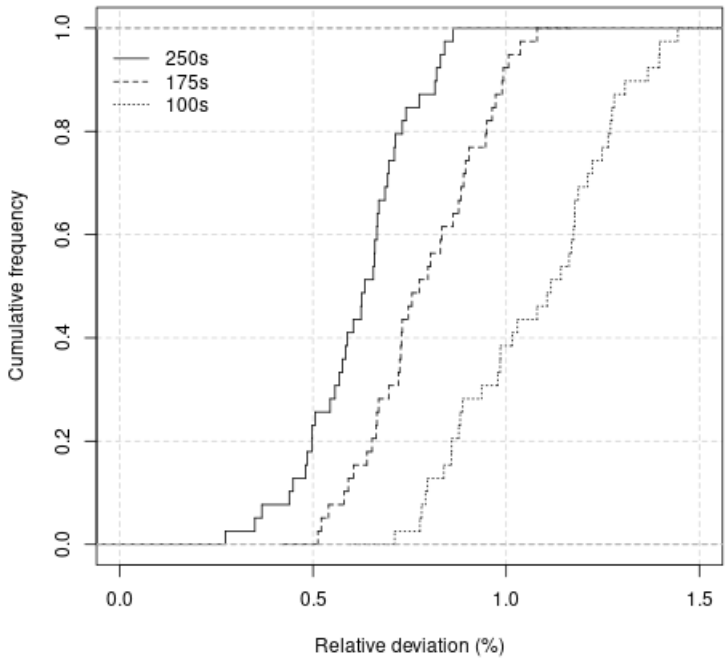

Figure 5: Solution quality distribution for the MA

Iterated Local Search (ILS)
The results on $n = 150$ instances are shown in the Table 8.

Algorithm	Δ_{avg} (%)	t_{avg} (s)	It_{avg}
ILS	0.402	250.541	271.205

Table 8: Iterated local search on instances of $n = 150$

The solution quality distribution (SQD) for 3 different runtime is presented at the Figure 6.

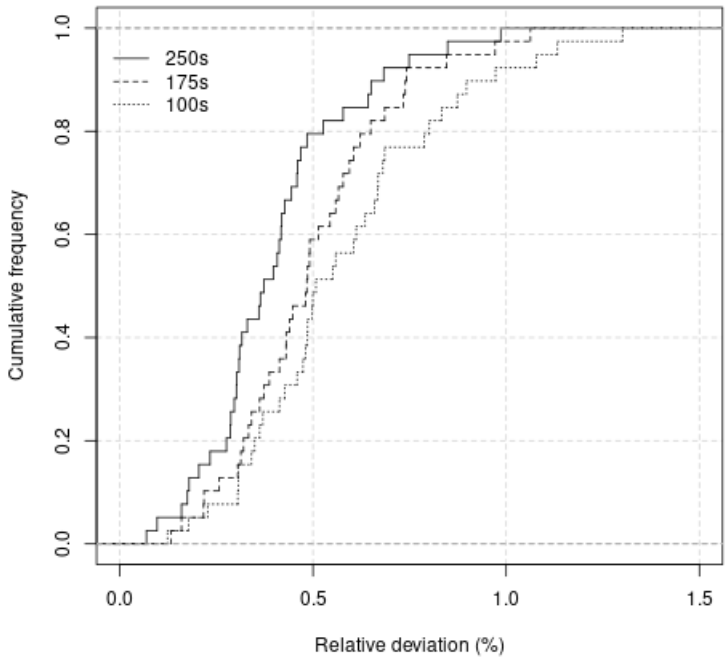

Figure 6: Solution quality distribution for the ILS

Comparison MA vs ILS

The correlation plot between the MA and ILS is presented at Figure 7. The correlation coefficient is 0.365. The standard error is $\sigma = 0.185$ and the equation of the linear regression is $y = 0.0851 + 0.513x$.

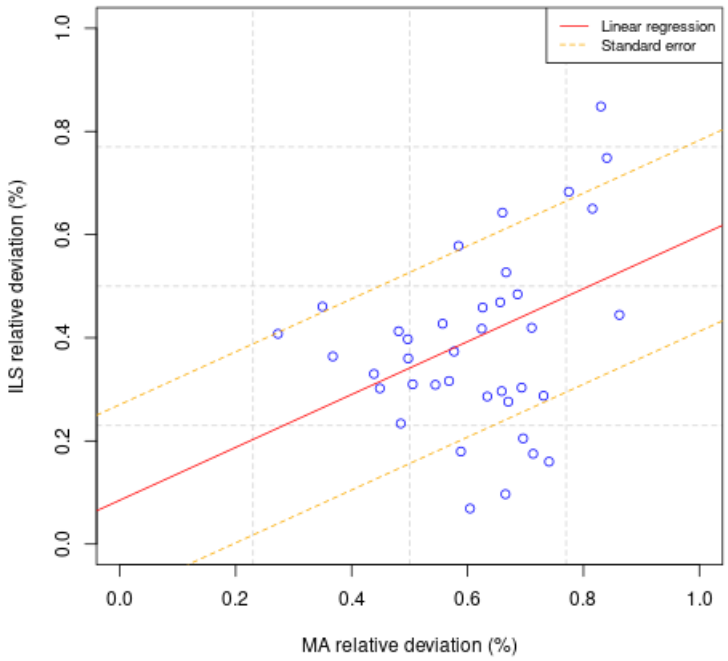

Figure 7: Correlation plot between MA and ILS

A comparison of during-runtime relative deviation on the N-be75eec_150 instance from the XLOLIB dataset during 1000 seconds is presented at the Figure 8.

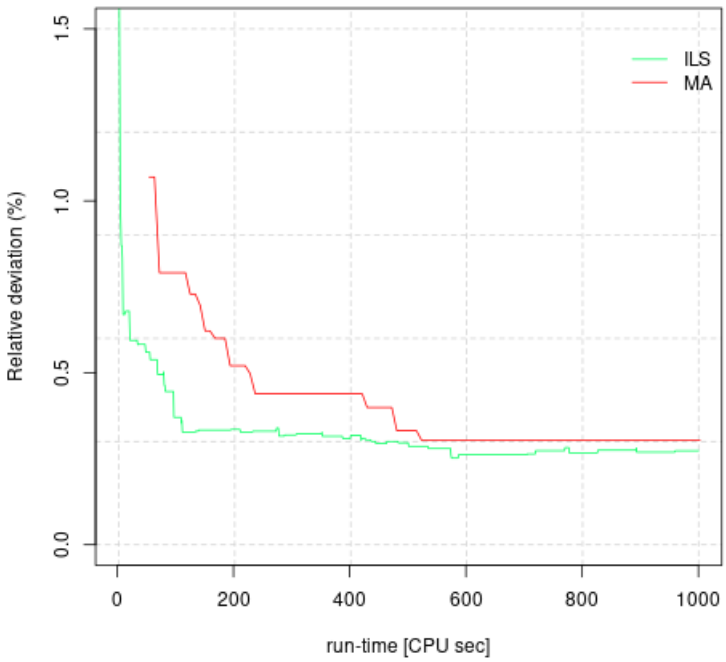

Figure 8: Comparison MA vs ILS

4.3 Statistical tests

In order to determine whether there is a statistical difference between the results of the different algorithms, 2 tests were carried out between them: the paired t-test (or Student's t-test) and the Wilcoxon signed rank test. The significance level α is put to $\alpha = 0.05$. The hypothesis that the distributions have the same average is rejected when p-value $< \alpha$. The test were made over the relative deviation from the best known solution.
The algorithm 1 is compared to the algorithm 2.

The results of the statistical tests on the different initial solution used (random and CW) are shown in the Table 9. We can conclude that the initial solution generated by the CW heuristic have a significant impact on the Best transpose, First transpose and First exchange.

Algorithm 1	Algorithm 2	T-test p-value	Wilcox p-value
Best exchange random	Best exchange CW	0.353	0.336
Best insert random	Best insert CW	0.131	0.180
Best transpose random	Best transpose CW	1.35e-25	3.638e-12
First exchange random	First exchange CW	0.010	0.018
First insert random	First insert CW	0.297	0.468
First transpose random	First transpose CW	1.103e-25	3.638e-12

Table 9: Statistical test on random and CW algorithms

The results of the statistical tests on the different iterative improvements (Best and First) are shown in the Table 10. We can conclude that the only algorithm where it makes a statistically difference is the transpose random algorithm.

Algorithm 1	Algorithm 2	T-test p-value	Wilcox p-value
Best exchange random	First exchange random	0.186	0.214
Best exchange CW	First exchange CW	0.690	0.644
Best transpose random	First transpose random	0.016	0.004
Best transpose CW	First transpose CW	0.084	0.028
Best insert random	First insert random	0.354	0.403
Best insert CW	First insert CW	0.514	0.624

Table 10: Statistical test on Best and First algorithms

The results of the statistical tests on the VND algorithms are shown in the Table 11. We can conclude from the Wilcoxon test that there isn't a significant statistical difference between the VND1 and VND2 relative deviations. But the order of evaluation of the different neighbourhood have an impact on the performance of the algorithm in terms of time.

Algorithm 1	Algorithm 2	T-test p-value	Wilcox p-value
VND1	VND2	0.784	0.591

Table 11: Statistical test on VND algorithms

The results of the statistical tests on the SLS algorithms are shown in the Table 12. We can deduce from the p-value that there's no significant difference between the MA and ILS algorithms.

Algorithm 1	Algorithm 2	Wilcox p-value
MA	ILS	4.723e-08

Table 12: Statistical test on SLS algorithms

5 Conclusion

To summarize what have been illustrated in the Section 4 (Results) :
For the classical methods (Best and First) the initial solution doesn't have a significant impact on the most of the algorithms.
Concerning the VND's algorithms, the order of evaluation of the neighbourhoods does have an impact on the execution time.
The best-time iterative improvement algorithm is the transpose pivoting rule algorithm, but as it's relative deviation is really high, the First exchange CW is the fastest one with decent results.
The most accurate iterative improvement algorithm, that have the lowest average relative deviation, is the First insert CW.
For the stochastic local search algorithms, the MA and ILS present both good results and we can't find a significant difference between them. But it is important to mention that the ILS find faster better solutions (as shown in Figure 8).

References

[MR11] Rafael Martí and Gerhard Reinelt. *The Linear Ordering Problem : Exact and heuristics methods*. en. 2011.

[MRD09] Rafael Martí, Gerhard Reinelt, and Abraham Duarte. *A benchmark library and a comparison of heuristic methods for the linear ordering problem*. en. 2009.

[SS04] Tommaso Schiavinotto and Thomas Stützle. *The Linear Ordering Problem: Instances, Search Space Analysis and Algorithms*. en. 2004.