ECOLE
POLYTECHNIQUE
DE BRUXELLES

MA1 - IRIF

HEURISTIC OPTIMIZATION

IMPLEMENTATION EXERCISE 1

---

https://github.com/iQuad427/PSPF_WT

Author

Quentin ROELS

Course

INFO-H413

Professor

Thomas STUTZLE

Christian CAMACHO VILLALON

Academic Year

2022-2023

Abstract

This report focus on solving the Permutation Flow-shop Scheduling Problem with Weighted Tardiness (PFSP-WT) using simple Iterative Improvement algorithms and comparing their efficiency and solution quality. The second part examine the results of two Variable Neighbourhood Descent algorithms and compare them with the previous results of II algorithms. Comparisons where made using the paired Wilcoxon test to show the statistical difference between the solutions of those algorithms.

Experimental results have shown that for single neighbourhoods algorithms solution quality mainly depends on the neighbourhood, the exchange one being the best one.

For the multiple neighbourhood algorithms tested, they were unexpectedly less good than expected, seemingly due to the chosen order of neighbourhoods causing the state to rapidly find a local optima far away from the best known-solution.

Table of contents

1 Introduction

1.1 Introduction

The Permutation Flow-shop Scheduling Problem (PFSP) is an active research subject with already many results. Many variants of this Operations Research classical problem exist, such as flow-time minimisation or total tardiness minimisation. One possible explanation of this popularity might be the direct application to the manufacturing industry and thus the underlying will to optimise the production lines. This problem being NP-Hard, there are no exact algorithms to solve it efficiently, allowing for continuous optimisation using new algorithms. Furthermore, due to the amount of existing results, it also makes it easy to compare the efficiency of different algorithms and the quality of the solution they provide.

1.2 Specifications

Focused on the Weighted Tardiness variant of the PFSP, the report investigate some of the simplest possible algorithm to optimise this kind of problem, i.e. iterative improvements algorithms. As stated earlier, the amount of previous results will allow for better comparison of the different algorithms solution. The paired Wilcoxon test has been used to show the statistical difference between the results using their deviation from the current best known solutions for given instances.

1.3 Objectives

This implementation exercise is divided in two parts.

- Implement simple iterative improvement algorithms to compare their solutions and execution. Those algorithms being combinations of different initialisation, pivoting and neighbourhood techniques, the goal is to see if a certain combination is better than the others.
- Implement two variable neighbourhood descent algorithms and compare them with their simple iterative improvement counterparts, to see if they lead to significantly better solutions.

1.4 Method

The 12 different iterative improvement and 4 variable neighbourhood descent algorithms were implemented using combinations of :

- 2 initialisation methods : random and simple RZ heuristic
- 2 pivoting rules : first improvement and best improvement
- 3 neighbourhood relations : exchange, insert and transpose

Note : see presentation slides for more details about those

As stated earlier, to show statistically the difference between the results of two algorithms, the paired Wilcoxon test is applied. All statistical tests are based on a null hypothesis that will either be rejected or not. For the Wilcoxon test, the null hypothesis is that "the median of the differences [between the two datasets] is zero". Thus, in this case, the test shows whether there is a significant difference between the results.

If the hypothesis is correctly rejected, it means that there is a statistical difference. This is why the test result in a p-value which indicates the probability of incorrectly rejecting the null hypothesis. This probability must be as low as possible to show that there is a difference.

Typically, a significance level α is chosen (a maximum allowable value) to compare the p-value with (here $\alpha = 0.05$); if the p-value is under this value, it means that this probability of falsely rejecting the null hypothesis is accepted.

In short, if a p-value is under 0.05 in this report, it means that, statistically, there is a significant difference between the values that are compared.

2 Iterative Improvement

2.1 Statistics

Hereunder are two tables containing different statistics on the 12 algorithms that were implemented for the exercise discriminated by the size of the instance they were tested on. Table 2.1 contains the statistics for all the algorithms for instance of size 50, while Table 2.2 contains the statistics for instance of size 100.

Note : two algorithms having the same specifications are on the same line.

Algorithm	Size	Deviation	Time (sec)
ii-best-rnd-ex	50	-29,9568	0,0981333
ii-best-rnd-ins	50	-4,89609	0,122618
ii-best-rnd-tran	50	271,414	0,0393053
ii-first-rnd-ex	50	-39,2156	0,322302
ii-first-rnd-ins	50	-21,6357	0,503399
ii-first-rnd-tran	50	264,235	0,0480356
ii-best-srz-ex	50	-30,8068	0,0906969
ii-best-srz-ins	50	-15,2437	0,0746060
ii-best-srz-tran	50	85,1007	0,0088318
ii-first-srz-ex	50	-40,7682	0,228158
ii-first-srz-ins	50	-26,7658	0,257870
ii-first-srz-tran	50	85,2780	0,0070813

Table 2.1: Algorithms statistics for 50 jobs

Algorithm	Size	Deviation	Time (sec)
ii-best-rnd-ex	100	-15,3481	1,06339
ii-best-rnd-ins	100	0,196022	1,05505
ii-best-rnd-tran	100	87,7223	0,616704
ii-first-rnd-ex	100	-23,1732	5,15766
ii-first-rnd-ins	100	-9,21661	6,35600
ii-first-rnd-tran	100	84,6680	0,792992
ii-best-srz-ex	100	-16,7051	0,482269
ii-best-srz-ins	100	-13,0212	0,425234
ii-best-srz-tran	100	0,428082	0,0680776
ii-first-srz-ex	100	-20,8418	0,848928
ii-first-srz-ins	100	-14,5708	0,618241
ii-first-srz-tran	100	0,356621	0,0590493

Table 2.2: Algorithms statistics for 100 jobs

2.2 Statistical Tests

This section discusses whether some algorithm components lead to better quality and/or better efficiency using the paired Wilcoxon statistical test to show if there is a statistically significant difference between the solutions.

2.2.1 Initial Solution

The goal of this experiment is to compare the initialisation rules, it is thus sufficient to consider one by one only the algorithms having the same specifications except for the initialisation rule.

For clarity and better understanding of the p-value result, the tables also contain the mean value of the deviation of each specification on the specific instance size.

SimpleRZ	SRZ.dev	Random	Random.dev	p-value
ii-best-srz-ex	-30,8068	ii-best-rnd-ex	-29,9568	0,695313
ii-best-srz-ins	-15,2437	ii-best-rnd-ins	-4,89609	0,193359
ii-best-srz-tran	85,1007	ii-best-rnd-tran	271,414	0,00195313
ii-first-srz-ex	-40,7682	ii-first-rnd-ex	-39,2156	0,160156
ii-first-srz-ins	-26,7658	ii-first-rnd-ins	-21,6357	0,00195313
ii-first-srz-tran	85,2780	ii-first-rnd-tran	264,235	0,00195313

Table 2.3: Comparison of the initialisation rules for instance of size 50

SimpleRZ	SRZ.dev	Random	Random.dev	p-value
ii-best-srz-ex	-16,7051	ii-best-rnd-ex	-15,3481	0,0644531
ii-best-srz-ins	-13,0212	ii-best-rnd-ins	0,196022	0,00195313
ii-best-srz-tran	0,428082	ii-best-rnd-tran	87,7223	0,00195313
ii-first-srz-ex	-20,8418	ii-first-rnd-ex	-23,1732	0,00195313
ii-first-srz-ins	-14,5708	ii-first-rnd-ins	-9,21661	0,00195313
ii-first-srz-tran	0,356621	ii-first-rnd-tran	84,6680	0,00195313

Table 2.4: Comparison of the initialisation rules for instance of size 100

For both instance sizes, the p-values show that there is a significant difference in the result when using different initialisation rules, except for the exchange neighbourhood. For the transpose and insert neighbourhood, the Simple RZ initialisation tends to have better solution quality.

From the statistical result and since no difference has been shown for the exchange neighbourhood, even if the exchange neighbourhood is chosen in the next section, the Simple RZ heuristic rule seem to be the best initialisation rule.

2.2.2 Pivoting Rule

As for the previous section, the comparison has to be done on the same specifications except for the part of the algorithm that is under investigation. The following tables are thus separated between best and first pivoting rule.

Specs	Best.dev	Best.time	First.dev	First.time	dev.p-value	time.p-value
ii-rnd-ex	-29,9568	0,0981333	-39,2156	0,322302	0,00195313	0,00195313
ii-rnd-ins	-4,89609	0,122618	-21,6357	0,503399	0,00195313	0,00195313
ii-rnd-tran	271,414	0,0393053	264,235	0,0480356	0,275391	0,0195313
ii-srz-ex	-30,8068	0,0906969	-40,7682	0,228158	0,00195313	0,00195313
ii-srz-ins	-15,2437	0,0746060	-26,7658	0,257870	0,00585938	0,00195313
ii-srz-tran	85,1007	0,00883182	85,2780	0,00708134	0,583882	0,193359

Table 2.5: Comparison of the pivoting rules for instance of size 50

Specs	Best.dev	Best.time	First.dev	First.time	dev.p-value	time.p-value
ii-rnd-ex	-15,3481	1,06339	-23,1732	5,15766	0,00195313	0,00195313
ii-rnd-ins	0,196022	1,05505	-9,21661	6,35600	0,00195313	0,00195313
ii-rnd-tran	87,7223	0,616704	84,6680	0,792992	0,130859	0,00390625
ii-srz-ex	-16,7051	0,482269	-20,8418	0,848928	0,00195313	0,00195313
ii-srz-ins	-13,0212	0,425234	-14,5708	0,618241	0,0488281	0,00390625
ii-srz-tran	0,428082	0,0680776	0,356621	0,0590493	0,674987	0,0371094

Table 2.6: Comparison of the pivoting rules for instance of size 100

For the deviation, the Wilcoxon test shows a statistical difference to the advantage of first improvement pivoting rule except for the algorithms using the transpose neighbourhood. This observation holds for both instance sizes.

However, for the execution time, the advantage clearly goes to the best improvement neighbourhood on both instances size. This is probably due to a faster convergence towards the optimal solution when going for the the highest improving solution.

2.2.3 Neighbourhood

This section compares the three neighbourhood rules, as a consequence, the comparison must be done for all pairs of neighbourhood to ensure correct observations. For better visualisation, the statistical result are divided in two; one sub-section for the deviation, and an other for the execution time.

Deviation

Specs	ins.dev	ex.dev	tran.dev	ins.ex.p-value	ins.tran.p-value	ex.tran.p-value
ii-best-rnd	-4,89609	-29,9568	271,414	0,00195313	0,00195313	0,00195313
ii-best-srz	-15,2437	-30,8068	85,1007	0,00585938	0,00195313	0,00195313
ii-first-rnd	-21,6357	-39,2156	264,235	0,00195313	0,00195313	0,00195313
ii-first-srz	-26,7658	-40,7682	85,2780	0,00585938	0,00195313	0,00195313

Table 2.7: Comparison of the solution quality neighbourhoods rules for instance of size 50

Specs	ins.dev	ex.dev	tran.dev	ins.ex.p-value	ins.tran.p-value	ex.tran.p-value
ii-best-rnd	0,196022	-15,3481	87,7223	0,00195313	0,00195313	0,00195313
ii-best-srz	-13,0212	-16,7051	0,428082	0,00585938	0,00195313	0,00195313
ii-first-rnd	-9,21661	-23,1732	84,6680	0,00195313	0,00195313	0,00195313
ii-first-srz	-14,5708	-20,8418	0,356621	0,00195313	0,00195313	0,00195313

Table 2.8: Comparison of the solution quality of the neighbourhood rules for instance of size 100

From the statistical tests result, it immediately follows that there is a clear ranking between the different neighbourhood rules, which is the same for both instance sizes:

Exchange > Insertion > Transpose.

As a matter of fact, when comparing by pairs we see that the exchange neighbourhood is better than all the others. Then, the insertion neighbourhood is better than the transpose neighbourhood and worst than the exchange neighbourhood. Finally, the transpose neighbourhood loses against all the others

Time

Specs	ins.time	ex.time	tran.time	ins.ex.p-value	ins.tran.p-value	ex.tran.p-value
ii-best-rnd	0,122618	0,0981333	0,0393053	0,00195313	0,00195313	0,00195313
ii-best-srz	0,0746060	0,090697	0,0088318	0,0371094	0,00195313	0,00195313
ii-first-rnd	0,503399	0,322302	0,0480356	0,00195313	0,00195313	0,00195313
ii-first-srz	0,257870	0,228158	0,0070813	0,105469	0,00195313	0,00195313

Table 2.9: Comparison of the execution time of the neighbourhoods rules for instance of size 50

Specs	ins.time	ex.time	tran.time	ins.ex.p-value	ins.tran.p-value	ex.tran.p-value
ii-best-rnd	1,05505	1,06339	0,616704	0,625000	0,00195313	0,00195313
ii-best-srz	0,425234	0,482269	0,0680776	0,232422	0,00195313	0,00195313
ii-first-rnd	6,35600	5,15766	0,792992	0,00195313	0,00195313	0,00195313
ii-first-srz	0,618241	0,848928	0,0590493	0,00195313	0,00195313	0,00195313

Table 2.10: Comparison of the execution time of the neighbourhood rules for instance of size 100

For both instance sizes, the transpose neighbourhood has the lowest execution time of all. Then for instances of size 50, the insertion neighbourhood is slower than the exchange neighbourhood. However, for instances of size 100, the exchange neighbourhood tend to slower than the insertion neighbourhood (the difference between the two is less significant).

We can deduct that the exchange neighbourhood grows bigger than the insertion one, meaning that for bigger instances, the exchange neighbourhood will be slower, leading to this ranking :

Transpose > Insertion > Exchange.

3 Variable Neighbourhood Descent

3.1 Comparison with II algorithms

VND	Size	Deviation	Time	ins.improve	ex.improve
vnd-best-rnd-tran-ex-ins	50	-34,1890	0,277736	29,5733	5,92599
vnd-best-rnd-tran-ins-ex	50	-31,8161	0,363580	27,5512	2,74154
vnd-best-srz-tran-ex-ins	50	-35,6639	0,203165	22,5721	7,18641
vnd-best-srz-tran-ins-ex	50	-29,2593	0,197040	14,4490	-2,56342
vnd-first-rnd-tran-ex-ins	50	-37,1886	0,447119	19,6956	-3,43519
vnd-first-rnd-tran-ins-ex	50	-31,7998	0,594070	12,7572	-12,4919
vnd-first-srz-tran-ex-ins	50	-35,3642	0,248080	11,3151	-9,21925
vnd-first-srz-tran-ins-ex	50	-33,3451	0,344675	8,39968	-12,6304
vnd-best-rnd-tran-ex-ins	100	-16,6196	4,34319	16,6164	1,44838
vnd-best-rnd-tran-ins-ex	100	-13,3810	4,63591	13,4383	-2,33989
vnd-best-srz-tran-ex-ins	100	-17,4245	1,39865	4,97218	0,825950
vnd-best-srz-tran-ins-ex	100	-17,1769	1,43430	4,60941	0,519038
vnd-first-rnd-tran-ex-ins	100	-21,1600	8,83566	13,1041	-2,62463
vnd-first-rnd-tran-ins-ex	100	-13,9024	10,2188	5,12136	-12,0931
vnd-first-srz-tran-ex-ins	100	-20,6418	1,44698	7,05744	-0,250512
vnd-first-srz-tran-ins-ex	100	-16,0205	1,54953	1,67390	-6,09074

Table 3.1: Improvement of the VND algorithms over solution of previously defined II algorithms

This experiment shows that even though the VND algorithms implemented are typically better than the simple insertion neighbourhood algorithms, they seem to be less good than those with exchange neighbourhood.

It is also possible to see some other behaviour in those results :

- transpose/exchange/insert neighbourhoods tend to have better results than transpose/insert/exchange neighbourhood.
- the deviation results tend to be better when the VND is using the first improvement pivoting rule, but the improvement compared to simple neighbourhood, is better with the best improvement pivoting rule.
- VND results are closer to the simple exchange neighbourhood algorithms for bigger instances than they are for smaller instances

Those observations follow the fact that exchange neighbourhoods is typically better at finding good solutions than the insert and transpose ones. Furthermore, the quality increase caused by the first-improvement pivoting rule tend to demonstrate that slower convergence allows for better solution quality in this setting.

However, there is a lack of improvement compared to the simple exchange neighbourhood. It could mean that the transpose neighbourhood at the first place of the VND neighbourhood search tend to put the algorithm in a local optima. Which could explain the bad results of the more complex VND algorithms compared to simpler algorithms with exchange neighbourhood.

3.2 Comparison between VND algorithms

For the following comparisons, the same method as for comparing the iterative improvement algorithms has been used, the way the conclusions were drawn being exactly the same as before.

3.2.1 Initialisation

SimpleRZ	SRZ.dev	Random	Random.dev	p-value
vnd-best-srz-tran-ex-ins	-35,6639	vnd-best-rnd-tran-ex-ins	-34,1890	0,431641
vnd-best-srz-tran-ins-ex	-29,2593	vnd-best-rnd-tran-ins-ex	-31,8161	0,160156
vnd-first-srz-tran-ex-ins	-35,3642	vnd-first-rnd-tran-ex-ins	-37,1886	0,375000
vnd-first-srz-tran-ins-ex	-33,3451	vnd-first-rnd-tran-ins-ex	-31,7998	0,492188

Table 3.2: Comparison of the initialisation rules for instance of size 50

SimpleRZ	SRZ.dev	Random	Random.dev	p-value
vnd-best-srz-tran-ex-ins	-17,4245	vnd-best-rnd-tran-ex-ins	-16,6196	0,431641
vnd-best-srz-tran-ins-ex	-17,1769	vnd-best-rnd-tran-ins-ex	-13,3810	0,0371094
vnd-first-srz-tran-ex-ins	-20,6418	vnd-first-rnd-tran-ex-ins	-21,1600	0,232422
vnd-first-srz-tran-ins-ex	-16,0205	vnd-first-rnd-tran-ins-ex	-13,9024	0,0371094

Table 3.3: Comparison of the initialisation rules for instance of size 100

This experiment shows that both pivoting rules give substantially the same results for both instance sizes. Except for the tran-ins-ex neighbourhood that benefits from the simple RZ heuristic for bigger instances.

3.2.2 Pivoting

Best	Best.dev	First	First.dev	p-value
vnd-best-rnd-tran-ex-ins	-34,1890	vnd-first-rnd-tran-ex-ins	-37,1886	0,0273438
vnd-best-rnd-tran-ins-ex	-31,8161	vnd-first-rnd-tran-ins-ex	-31,7998	0,845703
vnd-best-srz-tran-ex-ins	-35,6639	vnd-first-srz-tran-ex-ins	-35,3642	0,921875
vnd-best-srz-tran-ins-ex	-29,2593	vnd-first-srz-tran-ins-ex	-33,3451	0,00390625

Table 3.4: Comparison of the pivoting rules for instance of size 50

Best	Best.dev	First	First.dev	p-value
vnd-best-rnd-tran-ex-ins	-16,6196	vnd-first-rnd-tran-ex-ins	-21,1600	0,00195313
vnd-best-rnd-tran-ins-ex	-13,3810	vnd-first-rnd-tran-ins-ex	-13,9024	0,275391
vnd-best-srz-tran-ex-ins	-17,4245	vnd-first-srz-tran-ex-ins	-20,6418	0,00585938
vnd-best-srz-tran-ins-ex	-17,1769	vnd-first-srz-tran-ins-ex	-16,0205	0,556641

Table 3.5: Comparison of the pivoting rules for instance of size 100

The only algorithms and instance size for which there is statistical difference are the one for which the first improvement pivoting rule is better. This experiment shows that the first improvement pivoting rule is statistically better than the best improvement rule.

3.2.3 Neighbourhood

ins.ex	ins.ex.dev	ex.ins	ex.ins.dev	p-value
vnd-best-rnd-tran-ins-ex	-31,8161	vnd-best-rnd-tran-ex-ins	-34,1890	0,193359
vnd-best-srz-tran-ins-ex	-29,2593	vnd-best-srz-tran-ex-ins	-35,6639	0,00195313
vnd-first-rnd-tran-ins-ex	-31,7998	vnd-first-rnd-tran-ex-ins	-37,1886	0,00195313
vnd-first-srz-tran-ins-ex	-33,3451	vnd-first-srz-tran-ex-ins	-35,3642	0,275391

Table 3.6: Comparison of the neighbourhood rules for instance of size 50

ins.ex	ins.ex.dev	ex.ins	ex.ins.dev	p-value
vnd-best-rnd-tran-ins-ex	-13,3810	vnd-best-rnd-tran-ex-ins	-16,6196	0,00195313
vnd-best-srz-tran-ins-ex	-17,1769	vnd-best-srz-tran-ex-ins	-17,4245	1,00000
vnd-first-rnd-tran-ins-ex	-13,9024	vnd-first-rnd-tran-ex-ins	-21,1600	0,00195313
vnd-first-srz-tran-ins-ex	-16,0205	vnd-first-srz-tran-ex-ins	-20,6418	0,00390625

Table 3.7: Comparison of the neighbourhood rules for instance of size 100

The only algorithms and instance size for which there is statistical difference are the one for which the transpose-exchange-insertion neighbourhood is better. This experiment shows that the "transpose, exchange, insertion" neighbourhood is statistically better than the "transpose, insertion, exchange" one.

4 Conclusion

4.1 Iterative Improvement

When comparing the different combinations of operators for the iterative improvement algorithm, the results tend to show that the combination of either of the two initialisation methods, followed by a first-improvement pivoting rules using the exchange neighbourhood will eventually lead to the best results. This conclusion can be corroborated using the Tables 2.1 and 2.2 where the best deviations measured were for the ii-first-srz-ex and ii-first-rnd-ex algorithms.

4.2 Variable Neighbourhood Descent

Unexpectedly, the results on the implementations of the VND algorithms did not surpass the results quality of the best combination for the Iterative Improvement algorithms. As stated earlier, this result could be explained either by the order of the neighbourhoods (starting with transpose neighbourhood) or by a flow in the implementation of the algorithm. Seeing the result and the complementary tests made on other order of neighbourhoods, it seems like starting with the transpose neighbourhood tend to lock the execution on a local optimum state, leading to worse solution quality than the best combination of the previous section algorithms.

When comparing the VND algorithms between each other, conclusions are quite direct. While the initial solution construction does not affect the results significantly, a clear advantage goes to the first-improvement method compared to the best-improvement pivoting rule for both instance sizes. The neighbourhood order see the advantage going to the transpose/exchange/insert order compared to the transpose/insert/exchange one. This advantage grows even more with the increasing size of the instances.