Heuristic Optimization : Assignment 1

Kenzo Clauw

April 4, 2017

Abstract

1 Run the code

1.1 Compilation

Run the make command in the folder of the project. ./Flowshop is the main script to run the program.

1.2 Arguments

Command line arguments are used to select the different algorithms and their instances. These arguments should be executed in the following order :

./flowshop instance improvement neighborhood initial

Table 1: Syntax of each command line argument.

Commands	Values	Summary
./flowshop	./flowshop	Mandatory argument containing the runtime script made by the build command
Instance	instance path	Path of a single instance.
	–all	Runs the algorithm on every single instance.
Improvement	–First	iterative first improvement selects the first improving candidate solution
	–Best	iterative best improvement evaluates every solution
Neighborhood	–Transpose	Swaps each neighbor at position i and i+1
	–Exchange	Swaps each neighbor at position i and j
	–Insert	Inserts each element at index i through the whole solution space
Initial	–srz	The initial solution is generated by the srz heuristic
	–Random	the initial solution is generated at random

1.3 Example scenario 1

./flowshop –all

This scenario will run every single algorithm on each instance.
The script will then output the runtime of each algorithm per instance in the results.runtime.txt file.
The score of each algorithm per instance compared to the best solution is contained in the results.score.txt file.

1.4 Example scenario 2

This scenario will run a single algorithm on a specific instance. The purpose of this scenario is to test the individual algorithms on certain instance.

./flowshop ./instances/50 –first –transpose –random

2 Code Structure

This section describes the implementation of assignment 1 written in C++ 11.
The application is an extension of the code provided by the teacher.
For each class, the description of each function is as follows :

1. **Flowshop**
   Parses the command line arguments to the application and determines each algorithm and instance combination.

2. **Run**
   Contains the code that is necessary to run the algorithms on each of the instances.

3. **Pfspinstance**
   Compared to the original project there are a few extra methods added. When creating the instance there will be an extra 2 dimensional lookup-table used by the Compute_WCT_from method. The purpose of the Compute_WCT_from method is to recompute the value of the WCT starting at a certain job.

4. **Solution**
   The solution class contains a single solution vector. Storage move left/right moves an element in the solution vector from a certain position i to the position j(right) or from position j to i (left). Most of these methods are related to navigation, the main reason behind this is to make an abstraction allowing us to move jobs without having to worry about the underlying vector.

5. **Neighborhood**
   Contains all of the algorithms that are necessary to retrieve the neighborhood of the iterative improvement algorithm. Each neighborhood algorithm returns a boolean to determine if an improvement can be made. Choose_neighborhood will then function as a strategy method that executes on of the possible neighborhood algorithms and return the improvement boolean. This implementation of the neighborhood makes it easy for the variable neighborhood descent algorithm to just run one of the neighborhood algorithms until an improvement is made.

   .

3 Results

3.1 Exercise 1

This section contains the average relative percentage of the algorithm scores compared to the best solution score.

Table 2: Average percentage deviation from the best known solutions

Algorithm	50 jobs	100 jobs	All
First Random Transpose	31.986687	39.415857	35.701272
First Random Exchange	1.988601	1.681809	1.835205
First Random Insert	2.474552	2.585864	2.530208
First Srz Transpose	8.867471	9.936982	9.402226
First Srz Exchange	2.405070	2.378449	2.391759
First Srz Insert	1.578144	2.174206	1.876175
Best Random Transpose	32.5355873	41.166533	36.851060
Best Random Exchange	3.987419	4.783399	4.385409
Best Random Insert	3.482139	4.383328	3.932733
Best Srz Transpose	8.679424	10.103119	9.391272
Best Srz Exchange	3.707166	4.255597	3.981381
Best Srz Insert	2.449823	3.405681	2.927752

Table 3: Average computation time for each number of jobs in seconds

Algorithm	50 jobs	100 jobs	All
First Random Transpose	0.0018666667	0.0266333333	0.0142500000
First Random Exchange	0.4950000000	12.3558333333	6.4254166667
First Random Insert	0.3066666667	6.6004666667	3.4535666667
First Srz Transpose	0.0012000000	0.0143000000	0.0077500000
First Srz Exchange	0.1896333333	3.8876333333	2.0386333333
First Srz Insert	0.2947333333	6.6719666667	3.4833500000
Best Random Transpose	0.0033000000	0.0335333333	0.0184166667
Best Random Exchange	0.1510666667	2.3577666667	1.2544166667
Best Random Insert	0.2591000000	4.1130000000	2.1860500000
Best Srz Transpose	0.0021333333	0.0190333333	0.0105833333
Best Srz Exchange	0.1025333333	1.5269666667	0.8147500000
Best Srz Insert	0.1660666667	2.6129333333	1.3895000000

We notice that the algorithms with the transpose neighborhood are the worst algorithms but they take less time to compute.
The algorithms with the best pivoting rule take the most time to complete, this is because they have to find every candidate solution in the entire neighborhood search space.
On average, the algorithms with the insert or exchange neighborhood have a better score but need more computational time, this is because these algorithms have quadratic performance compared to the linear performance of the transpose neighborhood. First Random Exchange is the best scoring algorithm, but also takes the longest time to complete.

3.2 Exercise 2

Table 4: Average percentage deviation from the best known solutions

Algorithm	50 jobs	100 jobs	All
First Random Vnd1	1.690726	1.969149	1.829937
First Srz Vnd1	1.982103	1.982103	2.406998
First Random Vnd2	2.017300	2.915175	2.466238
First Srz Vnd2	2.086412	2.796580	2.441496

Table 5: Average computation time for each number of jobs in seconds

Algorithm	50 jobs	100 jobs	All
First Random Vnd1	0.2193666667	3.8318000000	2.0255833333
First Srz Vnd1	0.1342000000	1.8801333333	1.0071666667
First Random Vnd2	0.1729333333	3.1281000000	1.6505166667
First Srz Vnd2	0.1324333333	2.5647666667	1.3486000000

Table 6: Percentage improvement over the usage of a single neighborh ood

Algorithm	Vnd1 Random	Vnd1 Srz	Vnd2 Random	Vnd2 Srz
First Random Transpose	99%	99%	99%	99%
First Random Exchange	-252%	-168%	-163%	-164%
First Random Insert	-88%	-43%	-41%	-41%
First Srz Transpose	99%	99%	99%	99%
First Srz Exchange	-11%	15%	16%	16%
First Srz Insert	-90%	-44%	-42%	-42%
Best Random Transpose	99%	99%	99%	99%
Best Random Exchange	31%	47%	48%	48%
Best Random Insert	-19%	9%	9%	10%
Best Srz Transpose	99%	99%	99%	99%
Best Srz Exchange	55%	66%	66%	66%
Best Srz Insert	24%	42%	43%	43%

In most cases VND is faster then the algorithms with a single neighborhood, however first random insert/exchange are the dominant algorithms. The VND algorithms are faster then the best algorithms using a single neighborhood, there is only a big difference in relative score with the first random insert algorithm.

4 Statistical tests exercise 1

When conducting a T-test the following assumptions should be made :

1. **Normal Distribution**
   The data in both of the samples is normally distributed. We can test the distribution by using a Shapiro hypothesis test with a confidence interval of 0.05.

2. **Similar Variance**
   In most cases it is possible to calculate the variance for both populations and determine if their is a difference. Equal variances can also be tested by using a F-test.

3. **Equal Data**
   The size of both distributions should be the same.

When the assumption of equal variance cannot be made, then the alternative is the Wilconson test.

4.1 Normal Distribution

We will test the normal distribution assumption in both datasets by using a Shapiro Normality test.

4.1.1 Shapiro test relative scores dataset

Table 7: Shapiro normality test relative scores with confidence of 0.05

Algorithm	Random	Srz
First Transpose	0.4116	0.1331
First Exchange	0.454	0.0986
First Insert	0.7774	0.07031
Best Transpose	0.3622	0.3252
Best Exchange	0.3106	0.729
Best Insert	0.3021	0.4731

The green colored values in the tables indicate that the P-value is greater then 0.05 and thus we accept the null hypothesis. We conclude that the relative scores are normally distributed.

4.1.2 Shapiro test computation time dataset

Table 8: Shapiro normality test computation time with confidence of 0.05

Algorithm	Random	Srz
First Transpose	6.078e-08	5.695e-07
First Exchange	6.078e-08	4.353e-08
First Insert	2.912e-09	2.886e-08
Best Transpose	8.503e-10	4.661e-07
Best Exchange	8.323e-08	1.223e-07
Best Insert	7.245e-10	1.032e-07

The computation dataset is not normally distributed, However it is still possible to get decent results.
When the sample size is large enough, the central limit theorem proves that the distribution of the mean of data from any distribution approaches the normal

distribution.
We will be using non-parametric tests for this dataset, because these tests work with different distributions.

4.2 Which initial solution is preferable?

We will split the relative score and computation time dataset into 2 separate populations based on the random and Srz initial solution.

4.2.1 Equal variance relative score

By using a Z-test, we can determine if the assumptions of equal variance is made. The Z-test assumes that the variance of 2 populations is the same when the ratio between these values is roughly equal to 1.

Hypothesis 1 *true ratio of variances is not equal to 1(same variance)*

Null Hypothesis 1 *true ratio of variances is not equal to 1(not the same variance)*

A higher P-value when performing a Z-test, indicates that the null hypothesis is true.

Table 9: Z-test difference in variance test

Algorithm	P-value
First Transpose	9.758e-07
First Exchange	0.007478
First Insert	4.084e-05
Best Transpose	2.568e-08
Best Exchange	0.001084
Best Insert	0.001472

All of the algorithms reject the null hypothesis, we conclude that the variance is not the same.

4.2.2 Wilcox paired T-test

Because the variance is not the same, we will use a Wilcox paired T-test.

Hypothesis 2 *There is no difference between the algorithms with random and srz initial solution.*

Null Hypothesis 2 *There is a statistical difference between the algorithms with random and Srz initial solution.*

Table 10: Wilcox Paired T-test

Algorithm A	Algorithm B	Wilcox P-value
First Transpose Random	First Transpose Srz	1.671329e-11
First Exchange Random	First Exchange Srz	1.982484e-07
First Insert Random	First Insert Srz	2.072226e-08
Best Transpose Random	Best Transpose Srz	1.671329e-11
Best Exchange Random	Best Exchange Srz	0.001608026
Best Insert Random	Best Insert Srz	5.907791e-09

Based on the P-values we can conclude that their is a statistical difference between both initial solutions.

4.3 Which pivoting rule generates better quality solutions and which is faster?

We will split the relative score and computation time dataset into 2 separate populations, based on the best and first pivoting rule.
The populations will be tested by using the following hypothesis :

Hypothesis 3 *There is no difference between the algorithms with first and best pivoting rules.*

Null Hypothesis 3 *There is a statistical difference between the algorithms with first and best pivoting rules.*

4.3.1 Equal variance test relative scores

Before determining the statistical test, we will test if the variance between both populations is the same by using a Z-test.

Table 11: Z-test difference in variance test

Algorithm	P-value
Transpose Random	0.3341
Transpose Srz	0.8296
Exchange Random	0.002188
Exchange Srz	0.0002616
Insert Random	9.392e-06
Insert Srz	0.0004352

4.3.2 Hypothesis test relative scores

We will use the Wilcox test on the algorithms with a different variance and the T-test on the other algorithms.

Table 12: Hypothesis testing with confidence of 0.05

Algorithm A	Algorithm B	Wilcox P-value	T-test P-value
First Transpose Random	Best Transpose Random		4.847919e-07
First Transpose Srz	Best Transpose Srz		0.8798656
First Exchange Random	Best Exchange Random	1.671329e-11	
First Exchange Srz	Best Exchange Srz	9.057981e-11	
First Insert Random	Best Insert Random	2.045479e-11	
First Insert Srz	First Insert Srz	5.836319e-10	

Most of the algorithms in the hypothesis testing have a small P-values thus we reject the null hypothesis. We conclude that there is a significant difference between the First and Best pivoting rule.

4.3.3 Hypothesis test computation time

Because our data is not normally distributed, we will use a wilcox test.

Table 13: Hypothesis testing with confidence of 0.05

Algorithm A	Algorithm B	Wilcox P-value
First Transpose Random	Best Transpose Random	2.354955e-08
First Transpose Srz	Best Transpose Srz	8.865947e-07
First Exchange Random	Best Exchange Random	1.671067e-11
First Exchange Srz	Best Exchange Srz	1.670019e-11
First Insert Random	Best Insert Random	3.394998e-10
First Insert Srz	Best Insert Srz	2.261459e-11

The null hypothesis is rejected, the computation time between first and best insert is not the same. These results are obviously correct, because the best improvement algorithms have to search the complete neighborhood space.

4.4 Which neighborhood generates better quality solution and what computation time is required to reach local optima?

We will split the relative scores dataset into 3 populations, based on the different neighborhoods. Because we are working with multiple populations, we will be working with different hypothesis tests.

4.4.1 Equal variance test relative scores

When comparing the variance between 3 populations, we will be using the Bartlett test which is an alternative to the 2 samples Z-test.

Table 14: Bartlett test with confidence of 0.05

Algorithm A	Algorithm B	Algorithm C	P-value
First Transpose Random	First Exchange Random	First Insert Random	2.2e-16
First Transpose Srz	First Exchange Srz	First Insert Srz	2.2e-16
Best Transpose Random	Best Exchange Random	Best Insert Random	2.2e-16
Best Transpose Srz	Best Exchange Srz	Best Insert Srz	6.496e-16

We can conclude that the null hypothesis gets rejected, the variance is not the same.

4.4.2 Kruskall Wallis Hypothesis test relative scores

The Kruskall Wallis test is the alternative of the Wilcox Test when working with multiple populations.

Table 15: Kruskall Wallis test with confidence of 0.05

Algorithm A	Algorithm B	Algorithm C	P-value
First Transpose Random	First Exchange Random	First Insert Random	2.2e-16
First Transpose Srz	First Exchange Srz	First Insert Srz	2.2e-16
Best Transpose Random	Best Exchange Random	Best Insert Random	2.2e-16
Best Transpose Srz	Best Exchange Srz	Best Insert Srz	2.2e-16

We reject the null hypothesis and conclude that there is a significant difference between the neighborhoods.

4.4.3 Kruskall Wallis Hypothesis test computation time

Table 16: Kruskall Wallis test with confidence of 0.05

Algorithm A	Algorithm B	Algorithm C	P-value
First Transpose Random	First Exchange Random	First Insert Random	2.2e-16
First Transpose Srz	First Exchange Srz	First Insert Srz	2.2e-16
Best Transpose Random	Best Exchange Random	Best Insert Random	2.2e-16
Best Transpose Srz	Best Exchange Srz	Best Insert Srz	2.2e-16

We reject the null hypothesis and conclude that there is a significant difference between the computation times of the algorithms with a certain neighborhood.

5 Statistical tests exercise 2

The statistical tests performed in on the variable neighborhood descent algorithms are the same as in part 1.

The order of the neighborhood is as follows :
Vnd1 = Transpose Exchange Insert
Vnd2 = Transpose Insert Exchange

5.1 Normal Distribution

We will test the normal distribution assumption in both datasets by using a Shapiro Normality test.

5.1.1 Shapiro test relative scores dataset

Table 17: Shapiro normality test relative scores with confidence of 0.05

Algorithm	Random	Srz
Transpose Exchange Insert	0.2475	0.2366
Transpose Insert Exchange	0.1103	0.1638

We conclude that the relative scores are normally distributed.

5.1.2 Shapiro test computation time dataset

Table 18: Shapiro normality test relative scores with confidence of 0.05

Algorithm	Random	Srz
Transpose Exchange Insert	8.22e-09	8.435e-07
Transpose Insert Exchange	2.832e-09	1.593e-07

We conclude that the computation times are not normally distributed.

5.2 Which initial solution is preferable?

5.2.1 Equal variance relative score

Table 19: Z-test difference in variance test

Algorithm	P-value
First Vnd1	6.755e-10
First Vnd2	0.1125

5.2.2 Hypothesis test relative score

Table 20: Wilcox Paired T-test

Algorithm A	Algorithm B	T-test P-value	Wilcox P-value
First Vnd1 Random	First Vnd1 Srz	6.532386e-08	
First Vnd2 Random	First Vnd2 Srz		0.9033216

There is a significant difference between the initial solution in the Variable Neighborhood Descent version 1. Variable Neighborhood Descent Version 2 is not affected by the initial solution.

5.3 Which neighborhood generates better quality solution and what computation time is required to reach local optima?

5.3.1 Equal variance test relative scores

Table 21: Z-test with confidence of 0.05

Algorithm A	Algorithm B	P-value
First Vnd1 Random	First Vnd2 Random	0.0002654
First Vnd1 Srz	First Vnd2 Srz	0.2308

5.3.2 Hypothesis test relative scores

Table 22: Hypothesis test with confidence of 0.05

Algorithm A	Algorithm B	P-value T-test	P-value Wilcox
First Vnd1 Random	First Vnd2 Random	1.013337e-15	
First Vnd1 Srz	First Vnd2 Srz		0.771217

There is a significant difference between the order of the neighborhoods when performing iterative improvement with a random start. Between the relative scores of the neighborhood order with the Rsz heuristic, there is not a significant difference.

5.3.3 Hypothesis test computation time

Table 23: Hypothesis test with confidence of 0.05

Algorithm A	Algorithm B	P-value
First Vnd1 Random	First Vnd2 Random	4.916098e-09
First Vnd1 Srz	First Vnd2 Srz	0.0003050896

We reject the null hypothesis and conclude that there is a significant difference between the computation times of the algorithms with a certain neighborhood.

6 Conclusion

We conclude that the algorithms with a more complicated neighborhood such as insert/exchange, have a better relative score compared to the best solutions, but take more time to compute. The algorithms using the best improvement as a pivoting rule, are the slowest.

The algorithms using variable neighborhood descent with a neighborhood order of Transpose Exchange Insert is faster in general but also performs in a decent amount of time.
It is obvious that performing multiple first improvements of different neighborhoods, is better then using best improvements with expensive computation time, these methods have a fast computation and have a decent relative score.