Problem solving project

Job Scheduling using a Genetic Algorithm

Lucian Trestioreanu & Nina Çaushi

* *Brief Introduction of the problem “ Minimum Makespan Minimization”*

Job scheduling is an optimization problem in which N tasks with certain processing times are assigned to M processing units. As it originally comes from industry, the tasks are called jobs and the processing units – machines. The problem can have multiple formulations and constrains, depending on the practical given situation. It can be solved using multiple algorithms, but for our exercise we are using the Genetic Algorithm.

In our case, each job has a different processing time, depending on the machine it is assigned to. A job can be scheduled only once, on a single machine, and the “makespan” is the execution time for a given chromosome (a given combination of assignments of jobs to machines). The goal is to find (one of) the best assignment(s) of the jobs to the machines such that the completion time of the last job (all jobs have finished processing) is minimized – this would be the minimum makespan.

The job scheduling problem is an NP-hard problem i.e. *non-deterministic polynomial-time hard*. In our case the total number of possible assignments would be 51216, and we try to solve this problem in an efficient manner using a genetic algorithm.

Using a Genetic Algorithm, an appropriate job scheduling is accomplished by creating “a population” of individuals - random combinations of assignments of jobs to machines, and afterwards iterating genetic operations on the individuals inspired from nature - like selection (tournament, roulette), use of individual “fitness” – the best survives, crossover – combinations between individuals to explore the search space, mutations – probabilistic change in the structure of an individual, etc. Through repeated iterations on the individuals inside the population (generations), in which the best survives (or the worst is discarded), a good solution is being found.

Worth noting that mutation is mostly used to provide exploration and cross-over to lead population to converge towards one of the good solutions found so far. As such, while cross-over tries to converge to a specific point, mutation tries to avoid convergence and to explore new areas.

* *Coding section*

The sample code received was outputting a binary sequence of 0/1 in the form of a one dimensional array, and the maximum execution time.

We had to modify the code in order to solve the minimum makespan scheduling problem of 512 tasks on 16 machines using this genetic algorithm by minimizing the execution time.

In the code there are comments that describe the changes made. The changes, step by step:

We first did the change **\*(A) in the Algorithm class** (for mutation) and **\*(B) in Chromosome class** (to generate integers 1…16 - the 16 machines).

Kept the old code, in order to later be able to see the comparisons and the output of the modifications; commented the unused parts of the code.

The execution number of steps can be changed as needed in **\*(C) in the “Exe” class**. For the tests and the Matlab stats we used a small number of steps, and then increased this number to a high number of iterations to find the lower bound for each instance.

Next, we create a new instance of the problem by opening the problem instance we want to process - **\*(D) in the Exe class**.

The **instance parser** reads the processing times from the .txt files in order to later compute population fitness.

In the **InstanceParser class - \*(E)** - swap all commas with null

At **InstanceParser Class - \*(F)** - remove all spaces, in order to display the file correctly.

In **InstanceParserClass \*(G)** - erase the first column numbers.

In **Population Class \*(H)** - change value of **“bestf”** to max (as worst case), to obtain the minimized result.

Create a new class - **“MyproblemMMSP**” and override the abstract method “**evaluate**” from Problem.java by using an integer table with 16 machines. Further, the maximum execution time of each gene is calculated in “**MyproblemMMSP” - \*(I).**

I have also created in the exe class a “For” loop, which cycles the code for n times to get different result samples for the statistical analysis – in our case, n=50. For getting the lower bound (run the code only once), the “n” parameter at **\*(k) in the Exe class** has to be set to 1.

**Evaluate** each individual: parse the individual, read the machine each job is assigned to, compute the processing time for each machine by adding the corresponding processing times for all jobs assigned to it. Compare the processing times for all the 16 machines, and find the maximum – which is the makespan for the given individual: **MyproblemMMSP \*(J).**

**Print the result** of n runs of the algorithm (used 1 and 50) and the parameters used: **\*(L) in Exe class** - print the one dimensional array - 50 length (the makespans obtained for 50 runs)

* *Description of your algorithms*

*o Parameters Tuning*

*o Operators (Crossover, Mutation, etc.)*

*o* ***Important! Please justify your implementation***

To solve the minimum makespan problem, the genetic algorithm was used. There are three main parameters in GA: crossover probability, mutation probability and population size. Also different search could be used (Local Search: Gradient, etc.). Crossover randomly selects one or two points in the parents’ chromosomes and interchanges the two parents’ chromosomes at this point(s) to produce two new offsprings in hopes of producing better chromosomes. A crossover probability is between 0.6 and 1. After crossover, a mutation might happen which means that part of the offsprings chromosome might change in order to introduce diversity within the population. The probability of mutating a particular bit is typically between 0.001 and 0.1. Population size says how many individuals are in the population.

For the instance: **u\_s\_hihi\_512\_16.txt** we only tuned the parameters Pm and Pc and ran 9 possible combinations. For the crossover probability we chose the values 0.6; 0.8 and 1 while for the mutation probability we chose the values 0.001953125; 0.00390625 and 9.765625E-4 i.e. 1/512, 2/512, 0.5/512.

For the instance: **u\_s\_lohi\_512\_16.txt** the population size was also changed. The population values were 512; 1024; 2048. For the mutation probability we used the values 0.001953125; 0.009765625 and 0.01953125 and for the crossover probability we still kept the values 0.6; 0.8 and 1. This means that 27 possible algorithms were tested.

We have used a single point crossover, binary tournament, and mutation. We have changed one individual per step, and insert it into the population instead of the worst individual currently into the population. The population is evaluated each step. So to compute the best individual, the code had to be run for an increased number of steps but also runs faster.

While we could have tried to write more advanced code, we tried to balance the time so we can also explore the effect of different parameters, and also see the effect of these parameters on extensive searches - not only on short and fast runs.

* *Use ANOVA or Non-parametric analysis to study the effect of tuned parameters in your*

*algorithms (regarding the performance) on two specific instances;* ***u\_s\_hihi\_512\_16.txt*** *and* ***u\_s\_lohi\_512\_16.txt*** *(You only have to perform statistical analysis on these two instances)*

* *Explain the outcome from statistical analysis*

We have used the Kruskal Wallis test instead of ANOVA because it does not need any additional requirements or checks for the data, like normal probability.

However we have provided a normal distribution graph check for the first instance analysis we did.

- For the instance: **u\_s\_hihi\_512\_16.txt:**

We have tuned the parameters crossover and mutation probabilities, for three values each, which results in 9 possible algorithms.

Algorithm 1: pc=0.8, pm=0.001953125 (1/512)

Algorithm 2: pc=0.6, pm=0.001953125 (1/512)

Algorithm 3: pc=1.0, pm=0.001953125 (1/512)

Algorithm 4: pc=1.0, pm=0.00390625 (2/512)

Algorithm 5: pc=0.6, pm=0.00390625 (2/512)

Algorithm 6: pc=0.8, pm=0.00390625 (2/512)

Algorithm 7: pc=0.8, pm=9.765625E-4 (0.5/512)

Algorithm 8: pc=0.6, pm=9.765625E-4 (0.5/512)

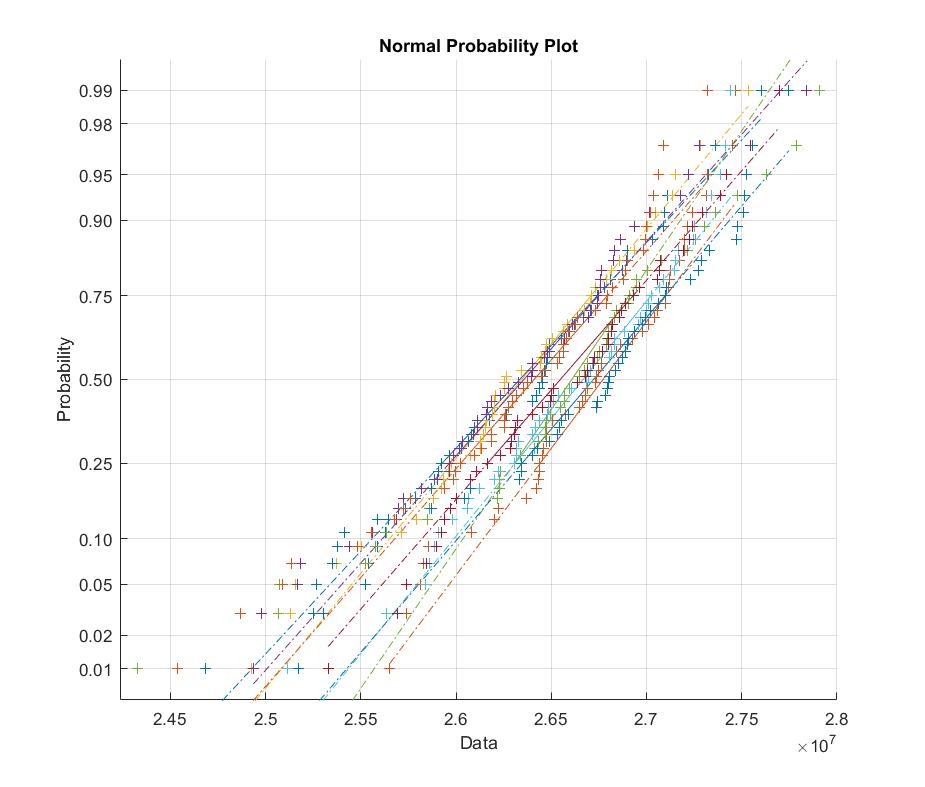
Algorithm 9: pc=1.0, pm=9.765625E-4 (0.5/512)

We ran each algorithm 50 times and collected the following results.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Alg1 | Alg2 | Alg3 | Alg4 | Alg5 | Alg6 | Alg7 | Alg8 | Alg9 |
| 1 | 26594395 | 25827243 | 26342091 | 26401053 | 26692129 | 27252061 | 27386220 | 25268109 | 26730117 |
| 2 | 27033346 | 26691659 | 25133489 | 25686431 | 24325386 | 26317239 | 27072528 | 26551773 | 26188261 |
| 3 | 27364718 | 27041751 | 25504343 | 27218389 | 25069822 | 27342290 | 27192246 | 26904594 | 25939730 |
| 4 | 25889296 | 25813770 | 25871440 | 26833533 | 26814483 | 26833017 | 25328776 | 25415389 | 25989096 |
| 5 | 26474980 | 26745896 | 26672764 | 26693418 | 27364920 | 27010498 | 26768679 | 26046592 | 24866368 |
| 6 | 27095611 | 26664714 | 26476074 | 27280296 | 25848873 | 25112606 | 26004897 | 27230748 | 27062763 |
| 7 | 26493422 | 27119310 | 26506886 | 27842227 | 26594802 | 26438128 | 26890947 | 27509745 | 26993575 |
| 8 | 26897736 | 27100301 | 27281816 | 25904689 | 26744623 | 26326812 | 25926142 | 27514051 | 27000559 |
| 9 | 25965540 | 27171977 | 26202433 | 26462369 | 26752628 | 27086764 | 26801473 | 27332045 | 26901126 |
| 10 | 26066978 | 27451665 | 25371683 | 26829451 | 26472045 | 27415848 | 26818930 | 27098449 | 25678367 |
| 11 | 26454109 | 26927610 | 27137803 | 26110625 | 27480172 | 26538533 | 26292431 | 26937334 | 26024443 |
| 12 | 26438318 | 26730249 | 27151746 | 26183676 | 26306641 | 26685167 | 25939838 | 26403717 | 26983695 |
| 13 | 26451776 | 26682255 | 27006431 | 26327865 | 25578294 | 25860777 | 26231173 | 26851887 | 26790041 |
| 14 | 25926304 | 26807010 | 27533559 | 26759405 | 26796393 | 27443000 | 27066550 | 25378799 | 26563600 |
| 15 | 25631190 | 27240909 | 25978881 | 25182339 | 27910700 | 25908068 | 26723136 | 26346249 | 26294140 |
| 16 | 26273749 | 27467896 | 26704479 | 26747855 | 26908168 | 26486038 | 25844965 | 26807232 | 26132867 |
| 17 | 26742412 | 27127287 | 25157040 | 26273839 | 26800916 | 26707223 | 27239953 | 26488256 | 25763326 |
| 18 | 25589119 | 27008843 | 26709634 | 26571189 | 27257358 | 26419142 | 27227354 | 26073602 | 24538806 |
| 19 | 26711424 | 27317566 | 26562414 | 26937609 | 27307220 | 26606427 | 26511442 | 26531662 | 26532324 |
| 20 | 26872270 | 27318356 | 26157490 | 26589493 | 26568873 | 26697749 | 25896742 | 26335463 | 25090882 |
| 21 | 26466776 | 26458811 | 26190578 | 25723812 | 25525759 | 27150187 | 26310043 | 26743261 | 25556788 |
| 22 | 26177981 | 26472676 | 26157050 | 26706725 | 27214120 | 25633757 | 25971090 | 27022907 | 27023630 |
| 23 | 26399450 | 26080163 | 26731043 | 26763806 | 26236101 | 26309618 | 27293115 | 27106528 | 26589459 |
| 24 | 27107922 | 27215332 | 26995136 | 26481351 | 26945347 | 26063753 | 26075752 | 26543842 | 26887636 |
| 25 | 26266433 | 26440883 | 26200569 | 27016693 | 26834188 | 26957698 | 26107975 | 26737263 | 25484291 |
| 26 | 26780305 | 27197957 | 26787494 | 25964798 | 26470691 | 26830400 | 26681135 | 26872360 | 26273723 |
| 27 | 26029602 | 26452449 | 25936509 | 25985569 | 27004891 | 26200262 | 26322213 | 26346221 | 26795384 |
| 28 | 25539606 | 26646493 | 25883356 | 25440559 | 26495150 | 25885208 | 25742487 | 26992738 | 26259833 |
| 29 | 26488870 | 25652329 | 26253236 | 25561647 | 26410444 | 27068430 | 26930466 | 26799551 | 27035400 |
| 30 | 25907675 | 26865274 | 26813952 | 26398826 | 27632668 | 25981114 | 26867838 | 27471605 | 26187471 |
| 31 | 26622681 | 26220706 | 26127958 | 26092832 | 25634536 | 27317033 | 27548241 | 26783840 | 26319888 |
| 32 | 26898313 | 26476630 | 26450020 | 27180858 | 26635996 | 26886612 | 26307371 | 26592434 | 26449263 |
| 33 | 26424556 | 26842881 | 26142145 | 24935552 | 26568479 | 27167194 | 26639652 | 26796633 | 27089516 |
| 34 | 25870696 | 27060088 | 25714339 | 26111659 | 26647930 | 26403835 | 26821342 | 26980971 | 26637698 |
| 35 | 25735548 | 26473513 | 25794663 | 26864925 | 26341157 | 26998252 | 26962944 | 25350776 | 26625280 |
| 36 | 27326426 | 26203892 | 26130670 | 26760895 | 25858973 | 26969864 | 26075493 | 26427182 | 26376142 |
| 37 | 26808832 | 26739519 | 25962969 | 24978156 | 27785568 | 26059265 | 26164432 | 25586427 | 26353374 |
| 38 | 25789128 | 26421911 | 27049981 | 26399742 | 26908697 | 26815110 | 26399459 | 25871406 | 26566075 |
| 39 | 25522315 | 26507885 | 26639062 | 26234438 | 26792049 | 26220877 | 26497170 | 27523824 | 26253296 |
| 40 | 26740842 | 26765608 | 26444165 | 26741555 | 26228519 | 27146296 | 26727608 | 27555155 | 26877820 |
| 41 | 26096419 | 26981057 | 26074336 | 25819799 | 26548532 | 27247522 | 26860832 | 26902932 | 26545780 |
| 42 | 24686216 | 26369027 | 25921481 | 26161220 | 26672685 | 26125063 | 27422401 | 27747923 | 25134235 |
| 43 | 27090203 | 27199209 | 26208430 | 26018638 | 26434046 | 27023816 | 26454344 | 25253503 | 26254867 |
| 44 | 26669205 | 27114238 | 26472156 | 25964060 | 25072897 | 25840173 | 26288524 | 27289602 | 26470600 |
| 45 | 25302155 | 25741969 | 26860805 | 26561732 | 26475937 | 27387669 | 25693190 | 27479792 | 26822716 |
| 46 | 26080895 | 25854347 | 26587446 | 26610997 | 26228505 | 26315194 | 27058614 | 26837980 | 25997518 |
| 47 | 26477969 | 27102455 | 26266558 | 26164946 | 26843446 | 26715690 | 26800452 | 27065513 | 25722062 |
| 48 | 27601653 | 26573529 | 24923826 | 25697051 | 26949602 | 26479819 | 26688299 | 26890964 | 26095462 |
| 49 | 26554091 | 26436355 | 26935687 | 25168551 | 26216639 | 26804592 | 27698261 | 25174993 | 27320377 |
| 50 | 25644571 | 26434335 | 26256067 | 26034195 | 27080384 | 26512473 | 26500259 | 27274382 | 26577638 |

We processed the results in Matlab, with the Kruskal-Wallis test which tests the hypothesis that all samples have the same mean or not (meaning the sample data from each column come from the same distribution or not).

**[pval, kwtab, stats] = kruskalwallis([Alg1, Alg2, Alg3, Alg4, Alg5, Alg6, Alg7, Alg8, Alg9])**



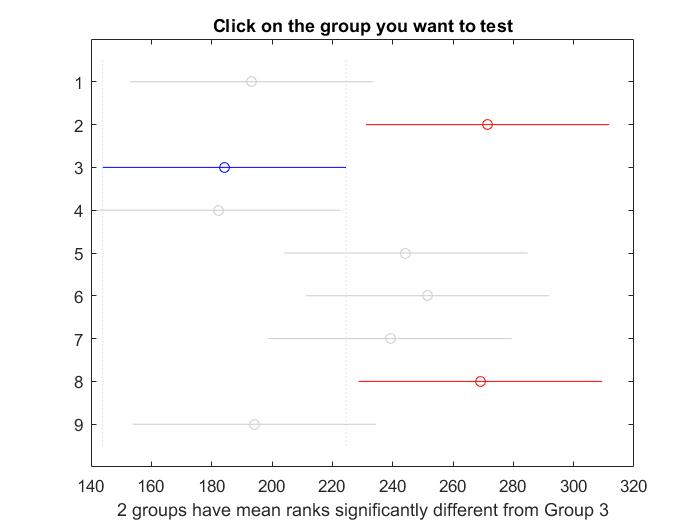
The Kruskal-Wallis test returns a stats structure which we further used to perform a follow-up multiple comparison test.

**Multcompare(stats)**

The p values obtained per pairs of algorithms:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Alg A | Alg B | **lower** | **mean difference** | **upper** | P Value |
| 1 | 2 | -158.8948 | -78.2200 | 2.4548 | 0.0658 |
| 1 | 3 | -71.6348 | 9.0400 | 89.7148 | 1.0000 |
| 1 | 4 | -69.6948 | 10.9800 | 91.6548 | 1.0000 |
| 1 | 5 | -131.8348 | -51.1600 | 29.5148 | 0.5666 |
| 1 | 6 | -139.0148 | -58.3400 | 22.3348 | 0.3776 |
| 1 | 7 | -126.5548 | -45.8800 | 34.7948 | 0.7061 |
| 1 | 8 | -156.5148 | -75.8400 | 4.8348 | 0.0849 |
| 1 | 9 | -81.5948 | -0.9200 | 79.7548 | 1.0000 |
| 2 | 3 | 6.5852 | 87.2600 | 167.9348 | 0.0226 |
| 2 | 4 | 8.5252 | 89.2000 | 169.8748 | 0.0176 |
| 2 | 5 | -53.6148 | 27.0600 | 107.7348 | 0.9820 |
| 2 | 6 | -60.7948 | 19.8800 | 100.5548 | 0.9978 |
| 2 | 7 | -48.3348 | 32.3400 | 113.0148 | 0.9469 |
| 2 | 8 | -78.2948 | 2.3800 | 83.0548 | 1.0000 |
| 2 | 9 | -3.3748 | 77.3000 | 157.9748 | 0.0727 |
| 3 | 4 | -78.7348 | 1.9400 | 82.6148 | 1.0000 |
| 3 | 5 | -140.8748 | -60.2000 | 20.4748 | 0.3331 |
| 3 | 6 | -148.0548 | -67.3800 | 13.2948 | 0.1904 |
| 3 | 7 | -135.5948 | -54.9200 | 25.7548 | 0.4652 |
| 3 | 8 | -165.5548 | -84.8800 | -4.2052 | 0.0304 |
| 3 | 9 | -90.6348 | -9.9600 | 70.7148 | 1.0000 |
| 4 | 5 | -142.8148 | -62.1400 | 18.5348 | 0.2898 |
| 4 | 6 | -149.9948 | -69.3200 | 11.3548 | 0.1604 |
| 4 | 7 | -137.5348 | -56.8600 | 23.8148 | 0.4147 |
| 4 | 8 | -167.4948 | -86.8200 | -6.1452 | 0.0239 |
| 4 | 9 | -92.5748 | -11.9000 | 68.7748 | 1.0000 |
| 5 | 6 | -87.8548 | -7.1800 | 73.4948 | 1.0000 |
| 5 | 7 | -75.3948 | 5.2800 | 85.9548 | 1.0000 |
| 5 | 8 | -105.3548 | -24.6800 | 55.9948 | 0.9901 |
| 5 | 9 | -30.4348 | 50.2400 | 130.9148 | 0.5916 |
| 6 | 7 | -68.2148 | 12.4600 | 93.1348 | 0.9999 |
| 6 | 8 | -98.1748 | -17.5000 | 63.1748 | 0.9991 |
| 6 | 9 | -23.2548 | 57.4200 | 138.0948 | 0.4005 |
| 7 | 8 | -110.6348 | -29.9600 | 50.7148 | 0.9661 |
| 7 | 9 | -35.7148 | 44.9600 | 125.6348 | 0.7289 |
| 8 | 9 | -5.7548 | 74.9200 | 155.5948 | 0.0934 |

P values < 0.05 mean that we have statistically significant differences with 95% confidence.



The results in the picture above indicate that there is a difference between algorithm 4 and 2 so the test rejects the hypothesis that the data in these two groups come from the same distribution. The same is true for algorithm 4 and 8. However, the test does not reject the hypothesis that algorithm 4 and the remaining six come from the same distribution. Therefore, these results suggest that the data from algorithms 1, 3, 4, 5, 6, 7, 9 come from the same distribution and the algorithms 2 and 8 come from a different distribution.

In other words, in some situations it is possible that Algorithm 1,3,4,5,6,7,9 will give same result, while, for sure, Alg2 and 8 will never give a similar result to Alg3&4. So if we look at the group Alg1,3,4,5,6,7,9, any of these could give a similar result, but because 3 and 4 have the best chances to give small makespans and in the same time different from Alg 2, 8, we decide that our best candidates are either Alg 3 or 4.

Based on the information gathered we have chosen to go ahead with algorithm 3 for this instance of the problem.

- For the instance: **u\_s\_lohi\_512\_16.txt:**

We tuned the parameters crossover, mutation probabilities, and population size for three values each, which results in 27 possible algorithms.

Alg1: pc=1.0, pm=0.001953125 (1/512), popsize=512

Alg2: pc=1.0, pm=0.001953125 (1/512), popsize=1024

Alg3: pc=1.0, pm=0.001953125 (1/512), popsize=2048

Alg4: pc=1.0, pm=0.009765625 (5/512), popsize=512

Alg5: pc=1.0, pm=0.009765625 (5/512), popsize=1024

Alg6: pc=1.0, pm=0.009765625 (5/512), popsize=2048

Alg7: pc=1.0, pm=0.01953125 (10/512), popsize=512

Alg8: pc=1.0, pm=0.01953125 (10/512), popsize=1024

Alg9: pc=1.0, pm=0.01953125 (10/512), popsize=2048

Alg10: pc=0.8, pm=0.001953125 (1/512), popsize=512

Alg11: pc=0.8, pm=0.001953125 (1/512), popsize=1024

Alg12: pc=0.8, pm=0.001953125 (1/512), popsize=2048

Alg13: pc=0.8, pm=0.009765625 (5/512), popsize=512

Alg14: pc=0.8, pm=0.009765625 (5/512), popsize=1024

Alg15: pc=0.8, pm=0.009765625 (5/512), popsize=2048

Alg16: pc=0.8, pm=0.01953125 (10/512), popsize=512

Alg17: pc=0.8, pm=0.01953125 (10/512), popsize=1024

Alg18: pc=0.8, pm=0.01953125 (10/512), popsize=2048

Alg19: pc=0.6, pm=0.001953125 (1/512), popsize=512

Alg20: pc=0.6, pm=0.001953125 (1/512), popsize=1024

Alg21: pc=0.6, pm=0.001953125 (1/512), popsize=2048

Alg22: pc=0.6, pm=0.009765625 (5/512), popsize=512

Alg23: pc=0.6, pm=0.009765625 (5/512), popsize=1024

Alg24: pc=0.6, pm=0.009765625 (5/512), popsize=2048

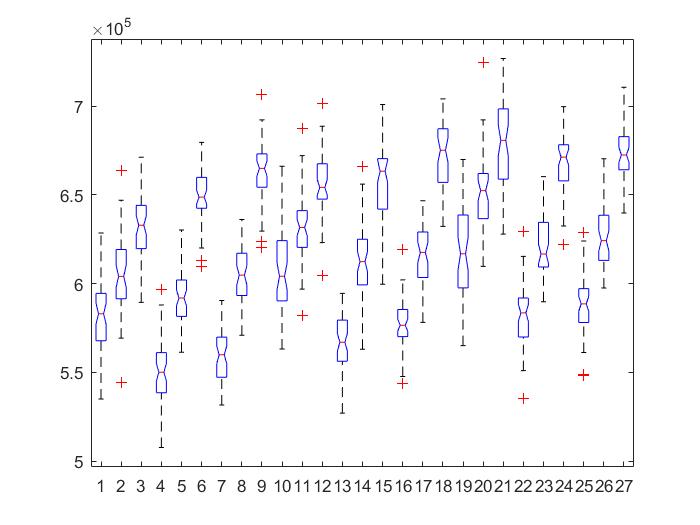
Alg25: pc=0.6, pm=0.01953125 (10/512), popsize=512

Alg26: pc=0.6, pm=0.01953125 (10/512), popsize=1024

Alg27: pc=0.6, pm=0.01953125 (10/512), popsize=2048

We ran each algorithm 50 times and collected the results which can be found in a separate Excel sheet: Project\_GA\Data\u\_s\_lohi\_512\_16\data\_u\_s\_lohi\_512\_16.xlsx

These results were further processed in Matlab, with the Kruskal-Wallis test like for the previous instance.



Next, we performed a follow-up multiple comparison test comparing each algorithm with one another. The results are shown in the table below.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **lower** | **mean difference** | **upper** | **P Value** | **A** | **B** | **lower** | **mean difference** | **upper** | **P Value** |
| 1 | 2 | -509.69359 | -221.48000 | 66.73359 | 0.46062 | 8 | 23 | -449.81359 | -161.60000 | 126.61359 | 0.94995 |
| 1 | 3 | -799.61359 | -511.40000 | -223.18641 | 0.00000 | 8 | 24 | -912.92359 | -624.71000 | -336.49641 | 0.00000 |
| 1 | 4 | -83.33359 | 204.88000 | 493.09359 | 0.63437 | 8 | 25 | -115.82359 | 172.39000 | 460.60359 | 0.90279 |
| 1 | 5 | -380.93359 | -92.72000 | 195.49359 | 0.99999 | 8 | 26 | -517.08359 | -228.87000 | 59.34359 | 0.38697 |
| 1 | 6 | -970.20359 | -681.99000 | -393.77641 | 0.00000 | 8 | 27 | -961.20359 | -672.99000 | -384.77641 | 0.00000 |
| 1 | 7 | -110.65359 | 177.56000 | 465.77359 | 0.87225 | 9 | 10 | 282.00641 | 570.22000 | 858.43359 | 0.00000 |
| 1 | 8 | -508.86359 | -220.65000 | 67.56359 | 0.46916 | 9 | 11 | 12.76641 | 300.98000 | 589.19359 | 0.02821 |
| 1 | 9 | -1,087.95359 | -799.74000 | -511.52641 | 0.00000 | 9 | 12 | -226.73359 | 61.48000 | 349.69359 | 1.00000 |
| 1 | 10 | -517.73359 | -229.52000 | 58.69359 | 0.38073 | 9 | 13 | 642.98641 | 931.20000 | 1,219.41359 | 0.00000 |
| 1 | 11 | -786.97359 | -498.76000 | -210.54641 | 0.00000 | 9 | 14 | 221.06641 | 509.28000 | 797.49359 | 0.00000 |
| 1 | 12 | -1,026.47359 | -738.26000 | -450.04641 | 0.00000 | 9 | 15 | -228.58359 | 59.63000 | 347.84359 | 1.00000 |
| 1 | 13 | -156.75359 | 131.46000 | 419.67359 | 0.99662 | 9 | 16 | 567.46641 | 855.68000 | 1,143.89359 | 0.00000 |
| 1 | 14 | -578.67359 | -290.46000 | -2.24641 | 0.04535 | 9 | 17 | 166.35641 | 454.57000 | 742.78359 | 0.00000 |
| 1 | 15 | -1,028.32359 | -740.11000 | -451.89641 | 0.00000 | 9 | 18 | -357.93359 | -69.72000 | 218.49359 | 1.00000 |
| 1 | 16 | -232.27359 | 55.94000 | 344.15359 | 1.00000 | 9 | 19 | 143.10641 | 431.32000 | 719.53359 | 0.00001 |
| 1 | 17 | -633.38359 | -345.17000 | -56.95641 | 0.00288 | 9 | 20 | -177.22359 | 110.99000 | 399.20359 | 0.99979 |
| 1 | 18 | -1,157.67359 | -869.46000 | -581.24641 | 0.00000 | 9 | 21 | -372.75359 | -84.54000 | 203.67359 | 1.00000 |
| 1 | 19 | -656.63359 | -368.42000 | -80.20641 | 0.00073 | 9 | 22 | 515.36641 | 803.58000 | 1,091.79359 | 0.00000 |
| 1 | 20 | -976.96359 | -688.75000 | -400.53641 | 0.00000 | 9 | 23 | 129.27641 | 417.49000 | 705.70359 | 0.00003 |
| 1 | 21 | -1,172.49359 | -884.28000 | -596.06641 | 0.00000 | 9 | 24 | -333.83359 | -45.62000 | 242.59359 | 1.00000 |
| 1 | 22 | -284.37359 | 3.84000 | 292.05359 | 1.00000 | 9 | 25 | 463.26641 | 751.48000 | 1,039.69359 | 0.00000 |
| 1 | 23 | -670.46359 | -382.25000 | -94.03641 | 0.00031 | 9 | 26 | 62.00641 | 350.22000 | 638.43359 | 0.00216 |
| 1 | 24 | -1,133.57359 | -845.36000 | -557.14641 | 0.00000 | 9 | 27 | -382.11359 | -93.90000 | 194.31359 | 0.99999 |
| 1 | 25 | -336.47359 | -48.26000 | 239.95359 | 1.00000 | 10 | 11 | -557.45359 | -269.24000 | 18.97359 | 0.10802 |
| 1 | 26 | -737.73359 | -449.52000 | -161.30641 | 0.00000 | 10 | 12 | -796.95359 | -508.74000 | -220.52641 | 0.00000 |
| 1 | 27 | -1,181.85359 | -893.64000 | -605.42641 | 0.00000 | 10 | 13 | 72.76641 | 360.98000 | 649.19359 | 0.00115 |
| 2 | 3 | -578.13359 | -289.92000 | -1.70641 | 0.04643 | 10 | 14 | -349.15359 | -60.94000 | 227.27359 | 1.00000 |
| 2 | 4 | 138.14641 | 426.36000 | 714.57359 | 0.00002 | 10 | 15 | -798.80359 | -510.59000 | -222.37641 | 0.00000 |
| 2 | 5 | -159.45359 | 128.76000 | 416.97359 | 0.99754 | 10 | 16 | -2.75359 | 285.46000 | 573.67359 | 0.05626 |
| 2 | 6 | -748.72359 | -460.51000 | -172.29641 | 0.00000 | 10 | 17 | -403.86359 | -115.65000 | 172.56359 | 0.99957 |
| 2 | 7 | 110.82641 | 399.04000 | 687.25359 | 0.00010 | 10 | 18 | -928.15359 | -639.94000 | -351.72641 | 0.00000 |
| 2 | 8 | -287.38359 | 0.83000 | 289.04359 | 1.00000 | 10 | 19 | -427.11359 | -138.90000 | 149.31359 | 0.99248 |
| 2 | 9 | -866.47359 | -578.26000 | -290.04641 | 0.00000 | 10 | 20 | -747.44359 | -459.23000 | -171.01641 | 0.00000 |
| 2 | 10 | -296.25359 | -8.04000 | 280.17359 | 1.00000 | 10 | 21 | -942.97359 | -654.76000 | -366.54641 | 0.00000 |
| 2 | 11 | -565.49359 | -277.28000 | 10.93359 | 0.07890 | 10 | 22 | -54.85359 | 233.36000 | 521.57359 | 0.34485 |
| 2 | 12 | -804.99359 | -516.78000 | -228.56641 | 0.00000 | 10 | 23 | -440.94359 | -152.73000 | 135.48359 | 0.97394 |
| 2 | 13 | 64.72641 | 352.94000 | 641.15359 | 0.00185 | 10 | 24 | -904.05359 | -615.84000 | -327.62641 | 0.00000 |
| 2 | 14 | -357.19359 | -68.98000 | 219.23359 | 1.00000 | 10 | 25 | -106.95359 | 181.26000 | 469.47359 | 0.84716 |
| 2 | 15 | -806.84359 | -518.63000 | -230.41641 | 0.00000 | 10 | 26 | -508.21359 | -220.00000 | 68.21359 | 0.47587 |
| 2 | 16 | -10.79359 | 277.42000 | 565.63359 | 0.07845 | 10 | 27 | -952.33359 | -664.12000 | -375.90641 | 0.00000 |
| 2 | 17 | -411.90359 | -123.69000 | 164.52359 | 0.99869 | 11 | 12 | -527.71359 | -239.50000 | 48.71359 | 0.29126 |
| 2 | 18 | -936.19359 | -647.98000 | -359.76641 | 0.00000 | 11 | 13 | 342.00641 | 630.22000 | 918.43359 | 0.00000 |
| 2 | 19 | -435.15359 | -146.94000 | 141.27359 | 0.98395 | 11 | 14 | -79.91359 | 208.30000 | 496.51359 | 0.59872 |
| 2 | 20 | -755.48359 | -467.27000 | -179.05641 | 0.00000 | 11 | 15 | -529.56359 | -241.35000 | 46.86359 | 0.27612 |
| 2 | 21 | -951.01359 | -662.80000 | -374.58641 | 0.00000 | 11 | 16 | 266.48641 | 554.70000 | 842.91359 | 0.00000 |
| 2 | 22 | -62.89359 | 225.32000 | 513.53359 | 0.42177 | 11 | 17 | -134.62359 | 153.59000 | 441.80359 | 0.97211 |
| 2 | 23 | -448.98359 | -160.77000 | 127.44359 | 0.95271 | 11 | 18 | -658.91359 | -370.70000 | -82.48641 | 0.00064 |
| 2 | 24 | -912.09359 | -623.88000 | -335.66641 | 0.00000 | 11 | 19 | -157.87359 | 130.34000 | 418.55359 | 0.99703 |
| 2 | 25 | -114.99359 | 173.22000 | 461.43359 | 0.89824 | 11 | 20 | -478.20359 | -189.99000 | 98.22359 | 0.77789 |
| 2 | 26 | -516.25359 | -228.04000 | 60.17359 | 0.39500 | 11 | 21 | -673.73359 | -385.52000 | -97.30641 | 0.00025 |
| 2 | 27 | -960.37359 | -672.16000 | -383.94641 | 0.00000 | 11 | 22 | 214.38641 | 502.60000 | 790.81359 | 0.00000 |
| 3 | 4 | 428.06641 | 716.28000 | 1,004.49359 | 0.00000 | 11 | 23 | -171.70359 | 116.51000 | 404.72359 | 0.99951 |
| 3 | 5 | 130.46641 | 418.68000 | 706.89359 | 0.00003 | 11 | 24 | -634.81359 | -346.60000 | -58.38641 | 0.00266 |
| 3 | 6 | -458.80359 | -170.59000 | 117.62359 | 0.91218 | 11 | 25 | 162.28641 | 450.50000 | 738.71359 | 0.00000 |
| 3 | 7 | 400.74641 | 688.96000 | 977.17359 | 0.00000 | 11 | 26 | -238.97359 | 49.24000 | 337.45359 | 1.00000 |
| 3 | 8 | 2.53641 | 290.75000 | 578.96359 | 0.04477 | 11 | 27 | -683.09359 | -394.88000 | -106.66641 | 0.00014 |
| 3 | 9 | -576.55359 | -288.34000 | -0.12641 | 0.04973 | 12 | 13 | 581.50641 | 869.72000 | 1,157.93359 | 0.00000 |
| 3 | 10 | -6.33359 | 281.88000 | 570.09359 | 0.06538 | 12 | 14 | 159.58641 | 447.80000 | 736.01359 | 0.00000 |
| 3 | 11 | -275.57359 | 12.64000 | 300.85359 | 1.00000 | 12 | 15 | -290.06359 | -1.85000 | 286.36359 | 1.00000 |
| 3 | 12 | -515.07359 | -226.86000 | 61.35359 | 0.40653 | 12 | 16 | 505.98641 | 794.20000 | 1,082.41359 | 0.00000 |
| 3 | 13 | 354.64641 | 642.86000 | 931.07359 | 0.00000 | 12 | 17 | 104.87641 | 393.09000 | 681.30359 | 0.00015 |
| 3 | 14 | -67.27359 | 220.94000 | 509.15359 | 0.46617 | 12 | 18 | -419.41359 | -131.20000 | 157.01359 | 0.99672 |
| 3 | 15 | -516.92359 | -228.71000 | 59.50359 | 0.38851 | 12 | 19 | 81.62641 | 369.84000 | 658.05359 | 0.00067 |
| 3 | 16 | 279.12641 | 567.34000 | 855.55359 | 0.00000 | 12 | 20 | -238.70359 | 49.51000 | 337.72359 | 1.00000 |
| 3 | 17 | -121.98359 | 166.23000 | 454.44359 | 0.93236 | 12 | 21 | -434.23359 | -146.02000 | 142.19359 | 0.98521 |
| 3 | 18 | -646.27359 | -358.06000 | -69.84641 | 0.00137 | 12 | 22 | 453.88641 | 742.10000 | 1,030.31359 | 0.00000 |
| 3 | 19 | -145.23359 | 142.98000 | 431.19359 | 0.98881 | 12 | 23 | 67.79641 | 356.01000 | 644.22359 | 0.00154 |
| 3 | 20 | -465.56359 | -177.35000 | 110.86359 | 0.87359 | 12 | 24 | -395.31359 | -107.10000 | 181.11359 | 0.99989 |
| 3 | 21 | -661.09359 | -372.88000 | -84.66641 | 0.00056 | 12 | 25 | 401.78641 | 690.00000 | 978.21359 | 0.00000 |
| 3 | 22 | 227.02641 | 515.24000 | 803.45359 | 0.00000 | 12 | 26 | 0.52641 | 288.74000 | 576.95359 | 0.04887 |
| 3 | 23 | -159.06359 | 129.15000 | 417.36359 | 0.99742 | 12 | 27 | -443.59359 | -155.38000 | 132.83359 | 0.96798 |
| 3 | 24 | -622.17359 | -333.96000 | -45.74641 | 0.00536 | 13 | 14 | -710.13359 | -421.92000 | -133.70641 | 0.00002 |
| 3 | 25 | 174.92641 | 463.14000 | 751.35359 | 0.00000 | 13 | 15 | -1,159.78359 | -871.57000 | -583.35641 | 0.00000 |
| 3 | 26 | -226.33359 | 61.88000 | 350.09359 | 1.00000 | 13 | 16 | -363.73359 | -75.52000 | 212.69359 | 1.00000 |
| 3 | 27 | -670.45359 | -382.24000 | -94.02641 | 0.00031 | 13 | 17 | -764.84359 | -476.63000 | -188.41641 | 0.00000 |
| 4 | 5 | -585.81359 | -297.60000 | -9.38641 | 0.03296 | 13 | 18 | -1,289.13359 | -1,000.92000 | -712.70641 | 0.00000 |
| 4 | 6 | -1,175.08359 | -886.87000 | -598.65641 | 0.00000 | 13 | 19 | -788.09359 | -499.88000 | -211.66641 | 0.00000 |
| 4 | 7 | -315.53359 | -27.32000 | 260.89359 | 1.00000 | 13 | 20 | -1,108.42359 | -820.21000 | -531.99641 | 0.00000 |
| 4 | 8 | -713.74359 | -425.53000 | -137.31641 | 0.00002 | 13 | 21 | -1,303.95359 | -1,015.74000 | -727.52641 | 0.00000 |
| 4 | 9 | -1,292.83359 | -1,004.62000 | -716.40641 | 0.00000 | 13 | 22 | -415.83359 | -127.62000 | 160.59359 | 0.99785 |
| 4 | 10 | -722.61359 | -434.40000 | -146.18641 | 0.00001 | 13 | 23 | -801.92359 | -513.71000 | -225.49641 | 0.00000 |
| 4 | 11 | -991.85359 | -703.64000 | -415.42641 | 0.00000 | 13 | 24 | -1,265.03359 | -976.82000 | -688.60641 | 0.00000 |
| 4 | 12 | -1,231.35359 | -943.14000 | -654.92641 | 0.00000 | 13 | 25 | -467.93359 | -179.72000 | 108.49359 | 0.85793 |
| 4 | 13 | -361.63359 | -73.42000 | 214.79359 | 1.00000 | 13 | 26 | -869.19359 | -580.98000 | -292.76641 | 0.00000 |
| 4 | 14 | -783.55359 | -495.34000 | -207.12641 | 0.00000 | 13 | 27 | -1,313.31359 | -1,025.10000 | -736.88641 | 0.00000 |
| 4 | 15 | -1,233.20359 | -944.99000 | -656.77641 | 0.00000 | 14 | 15 | -737.86359 | -449.65000 | -161.43641 | 0.00000 |
| 4 | 16 | -437.15359 | -148.94000 | 139.27359 | 0.98092 | 14 | 16 | 58.18641 | 346.40000 | 634.61359 | 0.00269 |
| 4 | 17 | -838.26359 | -550.05000 | -261.83641 | 0.00000 | 14 | 17 | -342.92359 | -54.71000 | 233.50359 | 1.00000 |
| 4 | 18 | -1,362.55359 | -1,074.34000 | -786.12641 | 0.00000 | 14 | 18 | -867.21359 | -579.00000 | -290.78641 | 0.00000 |
| 4 | 19 | -861.51359 | -573.30000 | -285.08641 | 0.00000 | 14 | 19 | -366.17359 | -77.96000 | 210.25359 | 1.00000 |
| 4 | 20 | -1,181.84359 | -893.63000 | -605.41641 | 0.00000 | 14 | 20 | -686.50359 | -398.29000 | -110.07641 | 0.00011 |
| 4 | 21 | -1,377.37359 | -1,089.16000 | -800.94641 | 0.00000 | 14 | 21 | -882.03359 | -593.82000 | -305.60641 | 0.00000 |
| 4 | 22 | -489.25359 | -201.04000 | 87.17359 | 0.67358 | 14 | 22 | 6.08641 | 294.30000 | 582.51359 | 0.03826 |
| 4 | 23 | -875.34359 | -587.13000 | -298.91641 | 0.00000 | 14 | 23 | -380.00359 | -91.79000 | 196.42359 | 0.99999 |
| 4 | 24 | -1,338.45359 | -1,050.24000 | -762.02641 | 0.00000 | 14 | 24 | -843.11359 | -554.90000 | -266.68641 | 0.00000 |
| 4 | 25 | -541.35359 | -253.14000 | 35.07359 | 0.19133 | 14 | 25 | -46.01359 | 242.20000 | 530.41359 | 0.26932 |
| 4 | 26 | -942.61359 | -654.40000 | -366.18641 | 0.00000 | 14 | 26 | -447.27359 | -159.06000 | 129.15359 | 0.95804 |
| 4 | 27 | -1,386.73359 | -1,098.52000 | -810.30641 | 0.00000 | 14 | 27 | -891.39359 | -603.18000 | -314.96641 | 0.00000 |
| 5 | 6 | -877.48359 | -589.27000 | -301.05641 | 0.00000 | 15 | 16 | 507.83641 | 796.05000 | 1,084.26359 | 0.00000 |
| 5 | 7 | -17.93359 | 270.28000 | 558.49359 | 0.10383 | 15 | 17 | 106.72641 | 394.94000 | 683.15359 | 0.00014 |
| 5 | 8 | -416.14359 | -127.93000 | 160.28359 | 0.99777 | 15 | 18 | -417.56359 | -129.35000 | 158.86359 | 0.99736 |
| 5 | 9 | -995.23359 | -707.02000 | -418.80641 | 0.00000 | 15 | 19 | 83.47641 | 371.69000 | 659.90359 | 0.00060 |
| 5 | 10 | -425.01359 | -136.80000 | 151.41359 | 0.99394 | 15 | 20 | -236.85359 | 51.36000 | 339.57359 | 1.00000 |
| 5 | 11 | -694.25359 | -406.04000 | -117.82641 | 0.00007 | 15 | 21 | -432.38359 | -144.17000 | 144.04359 | 0.98749 |
| 5 | 12 | -933.75359 | -645.54000 | -357.32641 | 0.00000 | 15 | 22 | 455.73641 | 743.95000 | 1,032.16359 | 0.00000 |
| 5 | 13 | -64.03359 | 224.18000 | 512.39359 | 0.43319 | 15 | 23 | 69.64641 | 357.86000 | 646.07359 | 0.00138 |
| 5 | 14 | -485.95359 | -197.74000 | 90.47359 | 0.70627 | 15 | 24 | -393.46359 | -105.25000 | 182.96359 | 0.99992 |
| 5 | 15 | -935.60359 | -647.39000 | -359.17641 | 0.00000 | 15 | 25 | 403.63641 | 691.85000 | 980.06359 | 0.00000 |
| 5 | 16 | -139.55359 | 148.66000 | 436.87359 | 0.98137 | 15 | 26 | 2.37641 | 290.59000 | 578.80359 | 0.04509 |
| 5 | 17 | -540.66359 | -252.45000 | 35.76359 | 0.19572 | 15 | 27 | -441.74359 | -153.53000 | 134.68359 | 0.97224 |
| 5 | 18 | -1,064.95359 | -776.74000 | -488.52641 | 0.00000 | 16 | 17 | -689.32359 | -401.11000 | -112.89641 | 0.00009 |
| 5 | 19 | -563.91359 | -275.70000 | 12.51359 | 0.08404 | 16 | 18 | -1,213.61359 | -925.40000 | -637.18641 | 0.00000 |
| 5 | 20 | -884.24359 | -596.03000 | -307.81641 | 0.00000 | 16 | 19 | -712.57359 | -424.36000 | -136.14641 | 0.00002 |
| 5 | 21 | -1,079.77359 | -791.56000 | -503.34641 | 0.00000 | 16 | 20 | -1,032.90359 | -744.69000 | -456.47641 | 0.00000 |
| 5 | 22 | -191.65359 | 96.56000 | 384.77359 | 0.99998 | 16 | 21 | -1,228.43359 | -940.22000 | -652.00641 | 0.00000 |
| 5 | 23 | -577.74359 | -289.53000 | -1.31641 | 0.04723 | 16 | 22 | -340.31359 | -52.10000 | 236.11359 | 1.00000 |
| 5 | 24 | -1,040.85359 | -752.64000 | -464.42641 | 0.00000 | 16 | 23 | -726.40359 | -438.19000 | -149.97641 | 0.00001 |
| 5 | 25 | -243.75359 | 44.46000 | 332.67359 | 1.00000 | 16 | 24 | -1,189.51359 | -901.30000 | -613.08641 | 0.00000 |
| 5 | 26 | -645.01359 | -356.80000 | -68.58641 | 0.00147 | 16 | 25 | -392.41359 | -104.20000 | 184.01359 | 0.99993 |
| 5 | 27 | -1,089.13359 | -800.92000 | -512.70641 | 0.00000 | 16 | 26 | -793.67359 | -505.46000 | -217.24641 | 0.00000 |
| 6 | 7 | 571.33641 | 859.55000 | 1,147.76359 | 0.00000 | 16 | 27 | -1,237.79359 | -949.58000 | -661.36641 | 0.00000 |
| 6 | 8 | 173.12641 | 461.34000 | 749.55359 | 0.00000 | 17 | 18 | -812.50359 | -524.29000 | -236.07641 | 0.00000 |
| 6 | 9 | -405.96359 | -117.75000 | 170.46359 | 0.99942 | 17 | 19 | -311.46359 | -23.25000 | 264.96359 | 1.00000 |
| 6 | 10 | 164.25641 | 452.47000 | 740.68359 | 0.00000 | 17 | 20 | -631.79359 | -343.58000 | -55.36641 | 0.00315 |
| 6 | 11 | -104.98359 | 183.23000 | 471.44359 | 0.83273 | 17 | 21 | -827.32359 | -539.11000 | -250.89641 | 0.00000 |
| 6 | 12 | -344.48359 | -56.27000 | 231.94359 | 1.00000 | 17 | 22 | 60.79641 | 349.01000 | 637.22359 | 0.00232 |
| 6 | 13 | 525.23641 | 813.45000 | 1,101.66359 | 0.00000 | 17 | 23 | -325.29359 | -37.08000 | 251.13359 | 1.00000 |
| 6 | 14 | 103.31641 | 391.53000 | 679.74359 | 0.00017 | 17 | 24 | -788.40359 | -500.19000 | -211.97641 | 0.00000 |
| 6 | 15 | -346.33359 | -58.12000 | 230.09359 | 1.00000 | 17 | 25 | 8.69641 | 296.91000 | 585.12359 | 0.03401 |
| 6 | 16 | 449.71641 | 737.93000 | 1,026.14359 | 0.00000 | 17 | 26 | -392.56359 | -104.35000 | 183.86359 | 0.99993 |
| 6 | 17 | 48.60641 | 336.82000 | 625.03359 | 0.00459 | 17 | 27 | -836.68359 | -548.47000 | -260.25641 | 0.00000 |
| 6 | 18 | -475.68359 | -187.47000 | 100.74359 | 0.79926 | 18 | 19 | 212.82641 | 501.04000 | 789.25359 | 0.00000 |
| 6 | 19 | 25.35641 | 313.57000 | 601.78359 | 0.01541 | 18 | 20 | -107.50359 | 180.71000 | 468.92359 | 0.85106 |
| 6 | 20 | -294.97359 | -6.76000 | 281.45359 | 1.00000 | 18 | 21 | -303.03359 | -14.82000 | 273.39359 | 1.00000 |
| 6 | 21 | -490.50359 | -202.29000 | 85.92359 | 0.66094 | 18 | 22 | 585.08641 | 873.30000 | 1,161.51359 | 0.00000 |
| 6 | 22 | 397.61641 | 685.83000 | 974.04359 | 0.00000 | 18 | 23 | 198.99641 | 487.21000 | 775.42359 | 0.00000 |
| 6 | 23 | 11.52641 | 299.74000 | 587.95359 | 0.02987 | 18 | 24 | -264.11359 | 24.10000 | 312.31359 | 1.00000 |
| 6 | 24 | -451.58359 | -163.37000 | 124.84359 | 0.94367 | 18 | 25 | 532.98641 | 821.20000 | 1,109.41359 | 0.00000 |
| 6 | 25 | 345.51641 | 633.73000 | 921.94359 | 0.00000 | 18 | 26 | 131.72641 | 419.94000 | 708.15359 | 0.00003 |
| 6 | 26 | -55.74359 | 232.47000 | 520.68359 | 0.35301 | 18 | 27 | -312.39359 | -24.18000 | 264.03359 | 1.00000 |
| 6 | 27 | -499.86359 | -211.65000 | 76.56359 | 0.56345 | 19 | 20 | -608.54359 | -320.33000 | -32.11641 | 0.01097 |
| 7 | 8 | -686.42359 | -398.21000 | -109.99641 | 0.00011 | 19 | 21 | -804.07359 | -515.86000 | -227.64641 | 0.00000 |
| 7 | 9 | -1,265.51359 | -977.30000 | -689.08641 | 0.00000 | 19 | 22 | 84.04641 | 372.26000 | 660.47359 | 0.00058 |
| 7 | 10 | -695.29359 | -407.08000 | -118.86641 | 0.00006 | 19 | 23 | -302.04359 | -13.83000 | 274.38359 | 1.00000 |
| 7 | 11 | -964.53359 | -676.32000 | -388.10641 | 0.00000 | 19 | 24 | -765.15359 | -476.94000 | -188.72641 | 0.00000 |
| 7 | 12 | -1,204.03359 | -915.82000 | -627.60641 | 0.00000 | 19 | 25 | 31.94641 | 320.16000 | 608.37359 | 0.01107 |
| 7 | 13 | -334.31359 | -46.10000 | 242.11359 | 1.00000 | 19 | 26 | -369.31359 | -81.10000 | 207.11359 | 1.00000 |
| 7 | 14 | -756.23359 | -468.02000 | -179.80641 | 0.00000 | 19 | 27 | -813.43359 | -525.22000 | -237.00641 | 0.00000 |
| 7 | 15 | -1,205.88359 | -917.67000 | -629.45641 | 0.00000 | 20 | 21 | -483.74359 | -195.53000 | 92.68359 | 0.72751 |
| 7 | 16 | -409.83359 | -121.62000 | 166.59359 | 0.99900 | 20 | 22 | 404.37641 | 692.59000 | 980.80359 | 0.00000 |
| 7 | 17 | -810.94359 | -522.73000 | -234.51641 | 0.00000 | 20 | 23 | 18.28641 | 306.50000 | 594.71359 | 0.02174 |
| 7 | 18 | -1,335.23359 | -1,047.02000 | -758.80641 | 0.00000 | 20 | 24 | -444.82359 | -156.61000 | 131.60359 | 0.96488 |
| 7 | 19 | -834.19359 | -545.98000 | -257.76641 | 0.00000 | 20 | 25 | 352.27641 | 640.49000 | 928.70359 | 0.00000 |
| 7 | 20 | -1,154.52359 | -866.31000 | -578.09641 | 0.00000 | 20 | 26 | -48.98359 | 239.23000 | 527.44359 | 0.29351 |
| 7 | 21 | -1,350.05359 | -1,061.84000 | -773.62641 | 0.00000 | 20 | 27 | -493.10359 | -204.89000 | 83.32359 | 0.63427 |
| 7 | 22 | -461.93359 | -173.72000 | 114.49359 | 0.89544 | 21 | 22 | 599.90641 | 888.12000 | 1,176.33359 | 0.00000 |
| 7 | 23 | -848.02359 | -559.81000 | -271.59641 | 0.00000 | 21 | 23 | 213.81641 | 502.03000 | 790.24359 | 0.00000 |
| 7 | 24 | -1,311.13359 | -1,022.92000 | -734.70641 | 0.00000 | 21 | 24 | -249.29359 | 38.92000 | 327.13359 | 1.00000 |
| 7 | 25 | -514.03359 | -225.82000 | 62.39359 | 0.41680 | 21 | 25 | 547.80641 | 836.02000 | 1,124.23359 | 0.00000 |
| 7 | 26 | -915.29359 | -627.08000 | -338.86641 | 0.00000 | 21 | 26 | 146.54641 | 434.76000 | 722.97359 | 0.00001 |
| 7 | 27 | -1,359.41359 | -1,071.20000 | -782.98641 | 0.00000 | 21 | 27 | -297.57359 | -9.36000 | 278.85359 | 1.00000 |
| 8 | 9 | -867.30359 | -579.09000 | -290.87641 | 0.00000 | 22 | 23 | -674.30359 | -386.09000 | -97.87641 | 0.00024 |
| 8 | 10 | -297.08359 | -8.87000 | 279.34359 | 1.00000 | 22 | 24 | -1,137.41359 | -849.20000 | -560.98641 | 0.00000 |
| 8 | 11 | -566.32359 | -278.11000 | 10.10359 | 0.07630 | 22 | 25 | -340.31359 | -52.10000 | 236.11359 | 1.00000 |
| 8 | 12 | -805.82359 | -517.61000 | -229.39641 | 0.00000 | 22 | 26 | -741.57359 | -453.36000 | -165.14641 | 0.00000 |
| 8 | 13 | 63.89641 | 352.11000 | 640.32359 | 0.00194 | 22 | 27 | -1,185.69359 | -897.48000 | -609.26641 | 0.00000 |
| 8 | 14 | -358.02359 | -69.81000 | 218.40359 | 1.00000 | 23 | 24 | -751.32359 | -463.11000 | -174.89641 | 0.00000 |
| 8 | 15 | -807.67359 | -519.46000 | -231.24641 | 0.00000 | 23 | 25 | 45.77641 | 333.99000 | 622.20359 | 0.00535 |
| 8 | 16 | -11.62359 | 276.59000 | 564.80359 | 0.08111 | 23 | 26 | -355.48359 | -67.27000 | 220.94359 | 1.00000 |
| 8 | 17 | -412.73359 | -124.52000 | 163.69359 | 0.99854 | 23 | 27 | -799.60359 | -511.39000 | -223.17641 | 0.00000 |
| 8 | 18 | -937.02359 | -648.81000 | -360.59641 | 0.00000 | 24 | 25 | 508.88641 | 797.10000 | 1,085.31359 | 0.00000 |
| 8 | 19 | -435.98359 | -147.77000 | 140.44359 | 0.98274 | 24 | 26 | 107.62641 | 395.84000 | 684.05359 | 0.00013 |
| 8 | 20 | -756.31359 | -468.10000 | -179.88641 | 0.00000 | 24 | 27 | -336.49359 | -48.28000 | 239.93359 | 1.00000 |
| 8 | 21 | -951.84359 | -663.63000 | -375.41641 | 0.00000 | 25 | 26 | -689.47359 | -401.26000 | -113.04641 | 0.00009 |
| 8 | 22 | -63.72359 | 224.49000 | 512.70359 | 0.43007 | 25 | 27 | -1,133.59359 | -845.38000 | -557.16641 | 0.00000 |
|  |  |  |  |  |  | 26 | 27 | -732.33359 | -444.12000 | -155.90641 | 0.00001 |

We can compare the algorithm 4 with the other algorithms to see which algorithms have the same distribution as our choice.

We can conclude that there are significant differences between algorithm 4 and the algorithms shown in red in the picture below.

Thus, the Kruskal-Wallis test rejects the hypothesis that these algorithms come from the same distributions. However, it does not reject the hypothesis that algorithm 4 and algorithms 1, 7, 13, 16, 22, 25 might come from the same distribution because their differences are less significant.



We have chosen algorithm 4 to be the best because it is clear from the above picture of distributions that it provides the lowest results and is statistically different from those algorithms that provide higher values, and statistically tens to provide the smallest means.

For obtaining the lower bound though, we have used a larger population size (1024/2048) and a high number of steps - ran the code for a long time in order to better explore the solution space.

On a second thought though I think that for a population size of 2048 we should have tried the algorithm 3 for finding the best chromosome because between the 2048 ones it seems to be the most appropriate.

Also I would have liked to investigate the issue of the population size effect during extended searches - longer runs of the code (many steps), when new better solutions occur less often.

* *Show average performance (from 50 independent runs) of your best algorithms on* ***all***

***instances*** *(If you have a lot of variations, state the parameter you use) e.g. Genetic Algorithm (Population: x, Crossover: x.xx%, Mutation: x.xx%, Local Search: Gradient, etc.)*

Average performance for the algorithms was run on the following setup, which seemed to tend to generally yield good results:

MakeSpans collected for 50 iterations - 100000 steps each, pc=1.0, pm=(5/512) , popsize=512.

The results are as follows:

|  |  |
| --- | --- |
| Instance | Average |
| u\_c\_hihi\_512\_16 | 15,966,430 |
| u\_c\_hilo\_512\_16 | 204,159 |
| u\_c\_lohi\_512\_16 | 529,462 |
| u\_c\_lolo\_512\_16 | 6,908 |
| u\_i\_hihi\_512\_16 | 16,662,829 |
| u\_i\_hilo\_512\_16 | 210,435 |
| u\_i\_lohi\_512\_16 | 540,188 |
| u\_i\_lolo\_512\_16 | 7,043 |
| u\_s\_hihi\_512\_16 | 15,757,344 |
| u\_s\_hilo\_512\_16 | 203,290 |
| u\_s\_lohi\_512\_16 | 481,462 |
| u\_s\_lolo\_512\_16 | 7,374 |

The table with data from all the 50 runs is in the annex \Project\_GA\Data\Average\_for\_50\_runs\_all\_instances\average\_for\_50\_runs\_all\_instances\_Excel.xlsx

*• Show the lower bound of your algorithm on* ***all instances*** *and present it in a similar*

*fashion to the file “LowerBound\_BestValue\_Instances”*

To obtain the lower bounds we ran the corresponding algorithms for a very long time on medium-large population sizes (1024/2048) in order to better explore the search space especially in the last part of the process, when improvements are hard to find.

The data, including the best chromosomes, was collected in the excel file: Project\_GA\Lower\_Bounds.xlsx

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instance** | **LB** | **Our Best** | **%** | **mean %** | **Comments: 1000000000 steps, pc/pm/popsize** |
| u\_c\_hihi\_512\_16.txt | 7346524 | 8478411 | 15.41% | 11.61% | pc=1.0, pm=5/512, popsize=512 |
| u\_c\_hilo\_512\_16.txt | 152700 | 167048 | 9.40% | pc=1.0, pm=5/512, popsize=2048 |
| u\_c\_lohi\_512\_16.txt | 238138 | 270916 | 13.76% | pc=1.0, pm=5/512, popsize=2048 |
| u\_c\_lolo\_512\_16.txt | 5132 | 5536 | 7.87% | pc=1.0, pm=5/512, popsize=1024 |
| u\_i\_hihi\_512\_16.txt | 2909326 | 3505029 | 20.48% | 24.07% | pc=1.0, pm=5/512, popsize=2048 |
| u\_i\_hilo\_512\_16.txt | 73057 | 90416 | 23.76% | pc=1.0, pm=5/512, popsize=2048 |
| u\_i\_lohi\_512\_16.txt | 101063 | 135997 | 34.57% | pc=1.0, pm=5/512, popsize=1024 |
| u\_i\_lolo\_512\_16.txt | 2529 | 2971 | 17.48% | pc=1.0, pm=5/512, popsize=1024 |
| u\_s\_hihi\_512\_16.txt | 4063563 | 5009057 | 23.27% | 20.11% | pc=1.0, pm=5/512, popsize=1024 |
| u\_s\_hilo\_512\_16.txt | 95419 | 110505 | 15.81% | pc=1.0, pm=5/512, popsize=1024 |
| u\_s\_lohi\_512\_16.txt | 120452 | 151330 | 25.64% | pc=1.0, pm=5/512, popsize=2048 |
| u\_s\_lolo\_512\_16.txt | 3414 | 3951 | 15.73% | pc=1.0, pm=5/512, popsize=2048 |

*Future Work*

*• Explain the future work if you were to have more time to improve the work and justify*

*your answer.*

If we were to have more time, we would like to:

- do a better research on the influence of population size on improving the search results when doing extended searches.

- study the effect of a crossover with two points cut which could yield better results

- implement and study the effect of a roulette versus the binary tournament.

- improve selection/acceptance implementation