Supplementary material to: MIP-based heuristics for the lot-scheduling problem in integrated pulp and paper production

Detailed parameter tuning procedure

1. Introduction

This document presents the procedures used during the parameter tuning for the presented solution methods. The parameter tuning was used for the developed methods and considering the problem of lot-scheduling problem applied to the pulp and paper industry.

2. Cplex parameters tuning

In the most recent versions, Cplex has an automatic parameter tuning function that considers its own hierarchy of parameters. This procedure is performed by using one or more instances of test. This procedure was tested here as a possible parameter tuning for Cplex. However, no set of parameters more efficient than the standard set was found, according to the metric of the automatic adjustment function. The difficulty encountered in its use is due to the way Cplex measures the efficiency of sets of parameters. The only metric used is computational time. All tests terminate due to exceeding the time limit and it was impossible to resolve the issue optimally. This generated equal metric values for all parameters sets, keeping the default values as the choose one. Tests were also carried out with a stopping condition defined from an acceptable gap that could be reached within the time limit by the best parameter set. However, the results obtained with the automatic parameter adjustment process were the same. For these reasons, it was decided to use a different parameter tuning methodology.

To perform the parameters tuning of Cplex version 12.10, ten instances were randomly selected. Thirteen parameters were selected to be adjusted, namely: use of the pre-solver, LP solution method used to solve the root node, cut limit, symmetry breaking intensity, emphasis used in the search process within the solution tree of the MIP, LP solution method used in other nodes, node selection strategy, branching direction, depth-first search strategy used, variable selection strategy for branching, probing level, use of local branching heuristic and application frequency of heuristics.

Parameter adjustments perform iteratively, considering the order of the 13 parameters described and fixing the best option for each parameter. Cplex was executed only once for each set of parameter values and instance test.

In the first iteration, the instances were solved with all parameters in default values. In the next iteration, the pre-solver was turned off and the other parameters have been maintained at their default values. Adjusting the first parameter, chosen between on and off, we move on to the next (LP solution method used to solve the root node), where all options presented in the manual were tested and only the best one was fixed for the following iterations. The process continues iteratively until all 13 parameters are adjusted. While tuning the Cplex parameters, the tests were time limited to 600 seconds.

As a result of the parameter tuning, changes were performed to 4 out of the 13 parameters considered. The variable selection strategy was chenged to strong branching (from the default value 0 to 3), which is indicated as advantageous for large and complicated problems. The level of probing has been changed to very aggressive (from the default value 0 to 3). MIP search emphasis has been changed to look for hidden solutions (from default value 0 to 4) and symmetry breaking has been turned off (from default value -1 to 0).

After parameter adjustment, all 540 instances were solved by Cplex with parameters in the adjusted value and the default value to compare the results and analyze the improvements obtained. We present the performance curves chart used to compare the two sets of parameters.

The Dolan & Moré (2002) (DM) performance curve chart aims to compare two or more solution methods according to time or solution quality. For exact methods, it can be used to compare the times used by each

one to prove the optimality of the solved problems. In the case of heuristic methods, it is particularly used to compare the solution quality between methods with similar solution times or limited to the same value. The comparison is made by solving a fixed set of instances for one or more solution methods.

Dolan & Moré (2002)'s chart works as follows. For each example, the solution with the lowest value (for minimization problems) is defined and the solution of each method is divided by this minimum value. This will result in a measure called τ, which indicates the proportion between the solution of each method and the best solution obtained. This measure is a relative deviation of the evaluated method solution concerning the best one found. Consider that p methods and s examples will be evaluated, and that r_{ps} indicates the relationship between the solution of p in example s and the best solution in this example ($r_{ps} = \frac{f_{ps}}{min\{f_{ps} \mid \forall p\}}$). The performance graph will indicate the percentage of examples of s in which the method p obtained a result of $r_{ps} \leq \tau$, for each value of τ defined on the abscissa axis. The percentage value with $\tau = 1$ indicates the number of best solutions obtained by the method. Furthermore, the moment a method reaches the limit of 100% indicates that this method did not have any results worse than the corresponding τ.

As can be seen in the graph in Figure 1, the Cplex with tuned parameters (Parameterized CPX) obtained better results than those obtained by the standard version. It occurred for the number of best solutions ($\tau = 1$) and for the rest of the chart since its performance curve is always closer to 100%. It means that it was possible to find better quality feasible integer solutions from the proper Cplex parameter tuning.

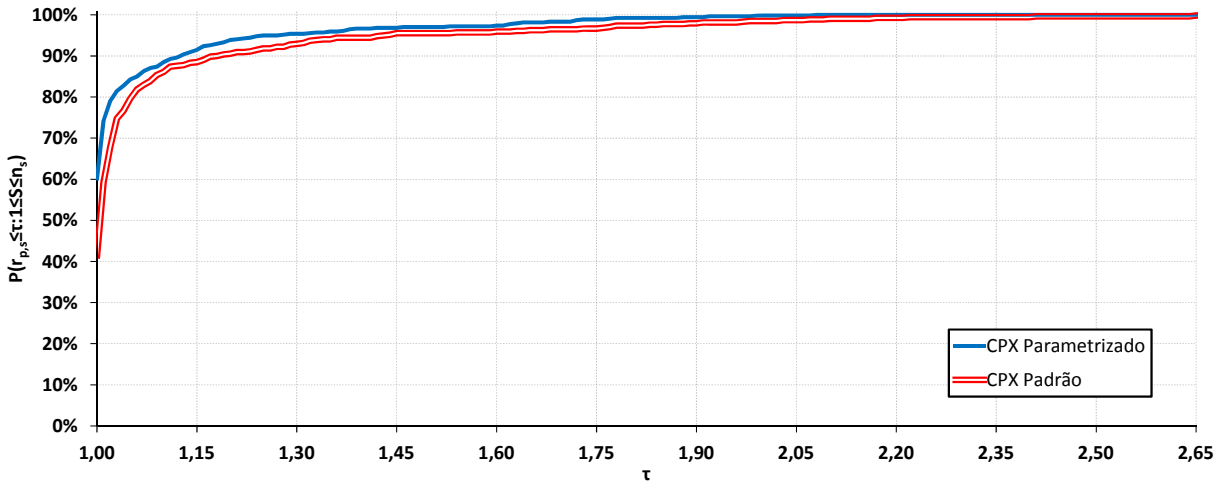

Figure 1: Performance curves chart of standard Cplex versus Cplex with tuned parameters (integer solutions only).

Figure 2 presents the gap of instances by the percentage deviation obtained in this method. The Gap of the instance represents the relative difference between the upper and lower bounds obtained by the solution method during the resolution of a test instance of the problem. The calculation of the Gap is obtained from the following formula:

$$Gap(x) = \frac{f(x) - LB}{f(x)} \cdot 100$$

where $f(x)$ represents the value of the objective function of the solution found and LB is the highest lower bound returned by all executed variations of Cplex after one hour of execution, namely: the standard Cplex, the parameterized Cplex and the Cplex with an initial solution of the constructive heuristics HCG and HCBC.

The performance curves chart in Figure 2 represents the percentage of instances in which each Cplex variant obtained a gap smaller than defined on the abscissa axis. For example, the tuned Cplex has a

percentage deviation equal to or smaller than 20% in almost 50% of the 540 copies. In these curves, we can see similar performance between the two variants of Cplex, each being better in part of the graph and none dominating the other completely. As the tuned version obtained better integer solutions, the explanation for such a similar percentage performance is that the parameterization somehow deteriorated the lower bound by modifying the MIP search emphasis or the other parameters.

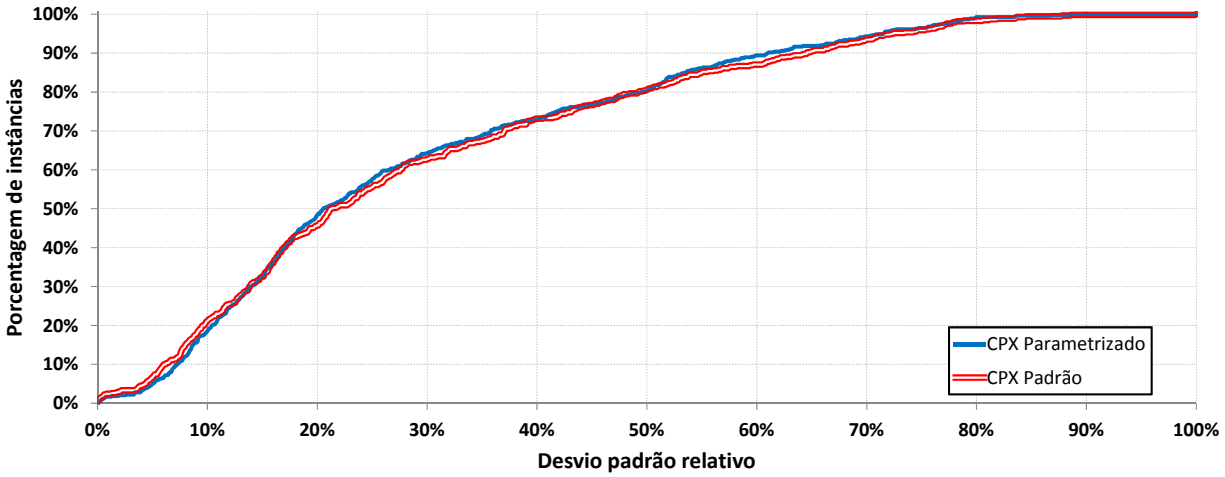

Figure 2: Performance curve chart of standard Cplex versus tuned Cplex (*gap*).

3. Parameters tuning of the developed methods

The parameter tuning of each method is performed according to its operating characteristics. There were methods in which whole set of instances was used, as in the case of the greedy constructive heuristic. As the heuristic generates feasible solutions on average in just under 1 second, testing a set of possible parameters for this constructive heuristic considering all examples does not become prohibitive due to computational time. On the other hand, the more complex relax-and-fix heuristic would take prohibitive time if all instances are considered.

For these more time-consuming methods, such as relax-and-fix and fix-and-optimize, a subset of 18 instances was randomly choosen as the parameter adjustment set. Each instance is chose from a group with different characteristics, which consists of a combination of the number of paper machines, grammages, and periods. In the case of sub-periods, only instances with size 4 were considered. For example, we have 15 instances with 1 paper machine ($M = 1$), 8 weights ($K = 8$), 7 planning periods ($T = 7$), and 4 sub-periods ($S = 4$. Of these 15 instances, only one was randomly drawn to be part of the parameter tuning set. As we generated 36 groups of instances and selected only the groups with the number of micro-periods equal to 4, we have 18 instances selected for parameterization. In this way it will be possible to have a subset that is reasonably representative of the set of instances generated. Even for solution approaches that include randomness, only one execution was considered for each test instance per approach and set of parameters in the parameter tuning process.

We present below the process used to adjust the parameters of each method and the set of parameters chosen for the final comparison. Analysis gaps are calculated using the best lower bound found after one hour of running the mathematical model with Cplex 12.10.

3.1. Greedy constructive heuristic (GCH)

The GCH has a fast execution, with an average of approximately 0.29 seconds for each instance. This reduced execution time helped the adjustment of the single parameter considering all 540 problem instances.

The α parameter defines the minimum demand necessary for an item to be allocated to a sub-period. To be more precise, it defines the proportion of the sub-period that will be occupied by the existing demand. The lower the value of α, the lower the demand required for a micro-period to be able to be allocated. In the case of $\alpha = 0$, for example, a positive demand for a grammage j is enough to allocate the production of j in the sub-period and machine.

Table 1 presents the minimum, average and maximum gap, and the average computational time for each value of α tested for all problem instances. Values from 0 to 1 with a step of 0.1 were tested. The results are similar and vary at most 1.44% in the average percentage deviation value. Time is even more similar, with an average ranging from 0.28 to 0.31 second. The four best average results, which are in the α range ranging from 0 to 0.3, were chosen for an analysis using a graph of performance curves.

Table 1: Gaps (%) and computational times (s) obtained by the GCH.

α	Minimum(%)	Average(%)	Maximum(%)	Average Time (s)
0.0	11.76	32.20	69.34	0.28
0.1	10.11	32.04	73.23	0.31
0.2	11.80	**31.81**	**69.01**	0.28
0.3	11.79	32.14	71.93	0.28
0.4	11.65	32.74	73.20	0.28
0.5	11.21	32.96	73.07	0.28
0.6	10.81	33.13	73.60	0.30
0.7	10.77	33.25	74.83	0.30
0.8	10.64	32.86	76.10	0.30
0.9	10.12	32.60	72.72	0.30
1.0	**9.76**	32.26	74.52	0.30
Total average	9.76	32.54	76.10	0.29

Figure 3 presents the graph with the performance curves (DM) of the four best adjustments of α (0 to 0.3). P0 indicates the performance curve with $\alpha = 0$, P1 refers to the performance of $\alpha = 0.1$, and so on.

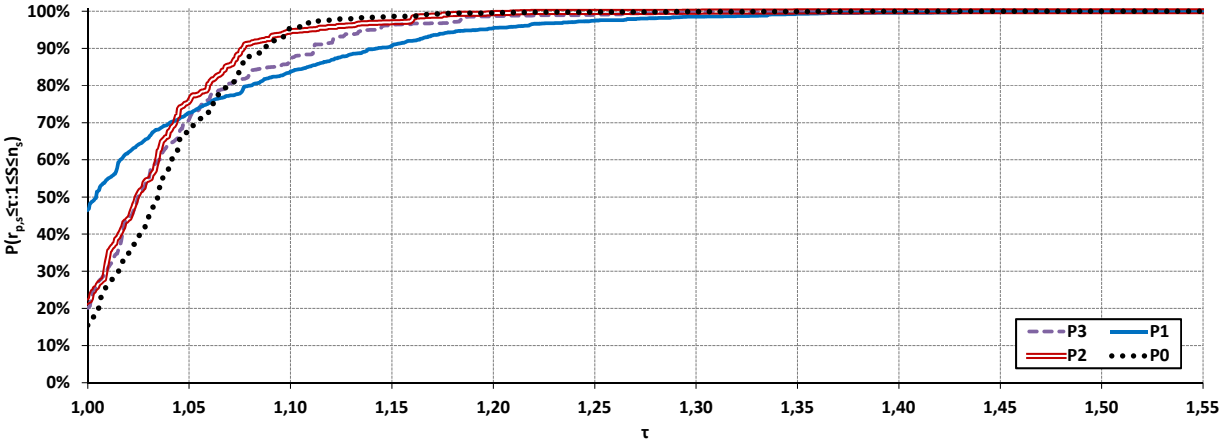

Figure 3: Graph of performance curves of the four best sets of greedy heuristic parameters.

We can see in this graph that the P1 heuristic has a higher number of better solutions than the other curves, finding the best solution in almost half of the cases. It is important to emphasize that the sum of the best solutions amount can be higher than 100%, as two or more methods can find the same solution, accounting for both. From $\tau = 1.043$, the P2 variant is superior, i.e., it has a higher percentage of results

with a maximum of 4.3% of worsening concerning the minimum solutions found. From $\tau = 1.064$ or 6.4%, P1 becomes the worst method. Even though it was not the best variation of α for $\tau > 1.05$, P1 obtained more than twice as many better solutions ($\tau = 0$) concerning the other variants. For this reason, the value chosen for α in this heuristic was $\alpha = 0.1$.

3.2. Production Cycle Based Heuristic (PCBH)

The PCBH parameter adjustment process was identical to the process applied to the GCH. As only the parameter α must be adjusted and the heuristic has reduced computational time, the whole set of instances was used to perform the parameter tuning.

Table 2 presents the average gaps considering each set of tests. In the case of $\alpha = 0.8$ we have a maximum gap of 100%, which indicates that there was at least one example in which the heuristic was not able to obtain a feasible solution. In the case of these tests, there was only one unsolved test sample among the 540 tests performed. For this reason, the parameter $\alpha = 0.8$ was not considered among the four best fits in the graph of Figure 4. The four parameters with the best average after $\alpha = 0.8$ were the values 0.1, 0.7, 0.9, and 1.0. The maximum difference of the average deviation of these parameters is equal to 0.36%, that is, the performance of the variants is more similar to the previous case.

Table 2: Gaps (%) and computational times (s) obtained by the PCBH.

α	Minimum(%)	Average(%)	Maximum(%)	Average Time (s)
0.0	11.05	34.12	63.56	2.07
0.1	**7.52**	29.13	69.72	1.16
0.2	10.77	33.19	61.21	2.06
0.3	10.24	32.06	60.37	2.06
0.4	8.54	30.87	**59.14**	2.05
0.5	9.40	30.07	63.89	2.05
0.6	9.35	29.50	60.54	2.06
0.7	8.86	28.81	60.94	2.05
0.8	10.14	**28.57**	100.00	2.06
0.9	10.14	28.77	65.77	1.16
1.0	7.62	29.11	68.64	1.16
Total average	7.52	30.38	100.00	1.81

Figure 4 presents the performance curves (DM) of the four best variants of the parameter α that did not have feasibility problems. We can see in this graph that the curves are very close and that the P7 variant ($\alpha = 0.7$) dominates an intermediate part of the graph, while the P9 variant ($\alpha = 0.9$) dominates from from $\tau \geq 1.09$. Again, the variant that had the highest number of best solutions was P1 ($\alpha = 0.1$). Due to the small difference with the best average variant and because it obtained more better solutions, the P1 variant was selected as the best parameter for this method. The value of $\alpha = 0.1$ can be related to the difference between the used inventory and delay costs of the tested instances. The cost of delay in the delivering of a ton of grammage j was considered equal to 10 times the cost of holding a ton of this same grammage. Therefore, requiring that at least 10% of the period ($\alpha = 0.1$) be used to meet demand seems reasonably in line with this cost relationship.

3.3. Fix-and-optimize heuristics

The parameter tuning of the fix-and-optimize heuristics was performmed from 18 instances chosen among the 540 used. By adjusting the parameters, you can define different partitionings to be applied sequentially, obtaining higher amount of possible combinations. In this case, we chose to apply three different types of partitioning in sequence and vary the size of the partitions of each type, to obtain all the analyzed combinations. Three distinct types of partitioning were used for the fix-and-optimize heuristics. Partitions were applied in the following order:

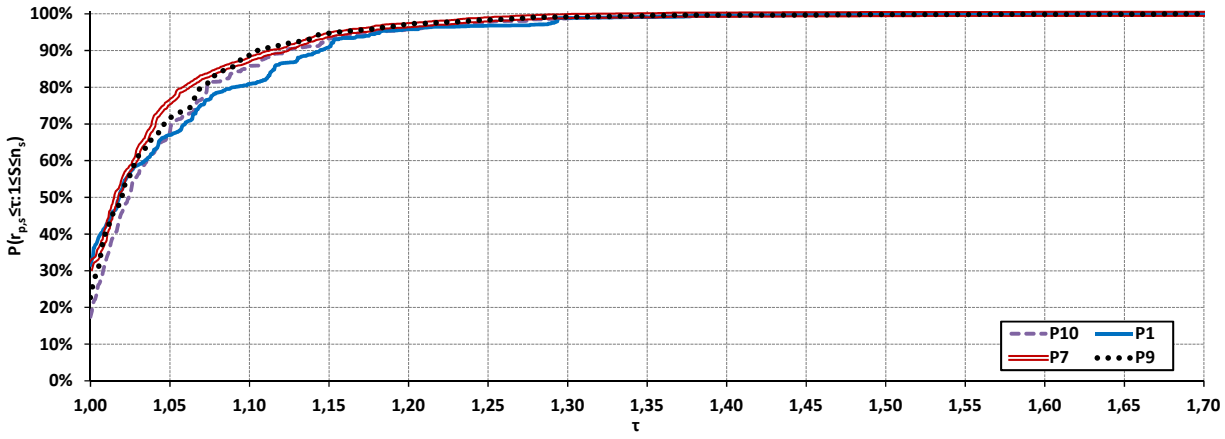

Figure 4: Graph of performance curves of the four best parameter sets of the PCBH.

1. Partitioning by sub-periods and machines (Partitioning 1)
2. Partitioning by Items and Machines (Partitioning 2)
3. Partitioning by items and periods (Partitioning 3)

These variations differ in the sizes of the three types of partitioning applied. Three size options were tested for partitioning 1, three size options for partitioning 2, and four sizes for partitioning 3. The combination of these options for each partitioning generated a total of 36 parameter variants, which are shown in Table 3. These partition sizes are used in the parameter tuning process for the remainder of this section.

As the fix-and-optimize heuristic is a solution improvement method, it needs an initial solution to start the procedure. For this task, GCH and PCBH are used to generate initial solutions. These two constructive heuristics generate results in reduced computational times. In addition, after parameter tuinng, both were able to produce feasible solutions for all instances. As a variation of the fix-and-optimize heuristic with modifications in the objective function was proposed, a complete comparison between the methods considering more than one type of initial solution was necessary. This is necessary to ensure that an initially constructed solution pattern is not influencing the evaluation of the improvement method. In addition, parameter tuning must be done with each of the method combinations to ensure a more adequate fit with the solution characteristics of each combination. Four solution combinations were generated between the constructive heuristics HCG and HCBC and the two improvement methods. The nomenclature used to differentiate these combinations was as follows:

- Traditional Fix-and-optimize combined with Greedy Constructive Heuristic (TFO-GCH)
- Traditional Fix-and-optimize combined with Production Cycle Based Heuristic (TFO-PCBH)
- Fix-and-optimize with variable objective function combined with Greedy Constructive Heuristic (VFO-GCH)
- Fix-and-optimize with variable objective function combined with Production Cycle Based Heuristic (VFO-PCBH)

The selection of the best variants was performed in two steps. In the first step, the choice of the five best options for analysis of the performance curves graph is made from the mean relative deviation (*gap*). In the second step, the variants with the best five averages are compared via performance curves (DM) graph. The results of the parameter tuning of these four combinations of methods are presented in sections 3.3.1 to 3.3.4, with only one set of parameters being selected for each combination of methods. For each of the eight combinations of constructive heuristics and fix-and-optimize improvement, 648 tests (18 instances and 36 parameter sets) limited to 3600 seconds each were performed.

Table 3: Parameter set of the 36 fix-and-optimize variants tested.

	Partitioning 1				Partitioning 2				Partitioning 3			
Variant	TF	TS	MF	MS	KF	KS	MF	MS	TF	TS	KF	KS
FO1	4	2	2	0	2	1	1	0	16	8	4	2
FO2	4	2	2	0	2	1	1	0	16	8	8	4
FO3	4	2	2	0	2	1	1	0	24	12	4	2
FO4	4	2	2	0	2	1	1	0	24	12	8	4
FO5	4	2	2	0	2	1	2	0	16	8	4	2
FO6	4	2	2	0	2	1	2	0	16	8	8	4
FO7	4	2	2	0	2	1	2	0	24	12	4	2
FO8	4	2	2	0	2	1	2	0	24	12	8	4
FO9	4	2	2	0	4	2	1	0	16	8	4	2
FO10	4	2	2	0	4	2	1	0	16	8	8	4
FO11	4	2	2	0	4	2	1	0	24	12	4	2
FO12	4	2	2	0	4	2	1	0	24	12	8	4
FO13	8	4	1	0	2	1	1	0	16	8	4	2
FO14	8	4	1	0	2	1	1	0	16	8	8	4
FO15	8	4	1	0	2	1	1	0	24	12	4	2
FO16	8	4	1	0	2	1	1	0	24	12	8	4
FO17	8	4	1	0	2	1	2	0	16	8	4	2
FO18	8	4	1	0	2	1	2	0	16	8	8	4
FO19	8	4	1	0	2	1	2	0	24	12	4	2
FO20	8	4	1	0	2	1	2	0	24	12	8	4
FO21	8	4	1	0	4	2	1	0	16	8	4	2
FO22	8	4	1	0	4	2	1	0	16	8	8	4
FO23	8	4	1	0	4	2	1	0	24	12	4	2
FO24	8	4	1	0	4	2	1	0	24	12	8	4
FO25	8	4	2	0	2	1	1	0	16	8	4	2
FO26	8	4	2	0	2	1	1	0	16	8	8	4
FO27	8	4	2	0	2	1	1	0	24	12	4	2
FO28	8	4	2	0	2	1	1	0	24	12	8	4
FO29	8	4	2	0	2	1	2	0	16	8	4	2
FO30	8	4	2	0	2	1	2	0	16	8	8	4
FO31	8	4	2	0	2	1	2	0	24	12	4	2
FO32	8	4	2	0	2	1	2	0	24	12	8	4
FO33	8	4	2	0	4	2	1	0	16	8	4	2
FO34	8	4	2	0	4	2	1	0	16	8	8	4
FO35	8	4	2	0	4	2	1	0	24	12	4	2
FO36	8	4	2	0	4	2	1	0	24	12	8	4

3.3.1. TFO-HCG Parameters tuning

Table 4 presents the average gaps obtained by the TFO-HCG heuristic considering each set of parameters. The five best parameter sets were highlighted in bold and selected for comparison in the performance curve graphs Overall average gap was 16.27%, and the maximum difference between the averages of the top five variants was 0.46%

Figure 5 presents the graph of performance curves of the best parameter sets of the TFO-HCG heuristic. In the chart, we have two sets of partitioning that stand out from the others, the variants FO17 and FO25. These two variants compete as best throughout the performance curve, with FO25 being best for τ values between 1 and $1,015$, while the FO17 variant dominates from $\tau = 1,018$ and reaches 100% of the copies in

Table 4: TFO-HCG variants average gap (%).

Variant	Gap	Variant	Gap	Variant	Gap	Variant	Gap
FO1	16.13	FO10	16.62	**FO19**	**15.70**	FO28	16.73
FO2	16.63	FO11	15.97	FO20	16.40	FO29	16.31
FO3	**15.76**	FO12	16.64	**FO21**	**15.81**	FO30	16.77
FO4	16.53	FO13	15.98	FO22	16.40	FO31	15.98
FO5	15.83	FO14	16.27	FO23	16.52	FO32	16.68
FO6	16.69	FO15	15.94	FO24	16.82	FO33	15.82
FO7	15.87	FO16	16.43	**FO25**	**15.71**	FO34	16.65
FO8	16.65	**FO17**	**15.35**	FO26	16.64	FO35	15.89
FO9	16.15	FO18	16.26	FO27	16.23	FO36	16.97

the first place, shortly after $\tau = 1,045$.

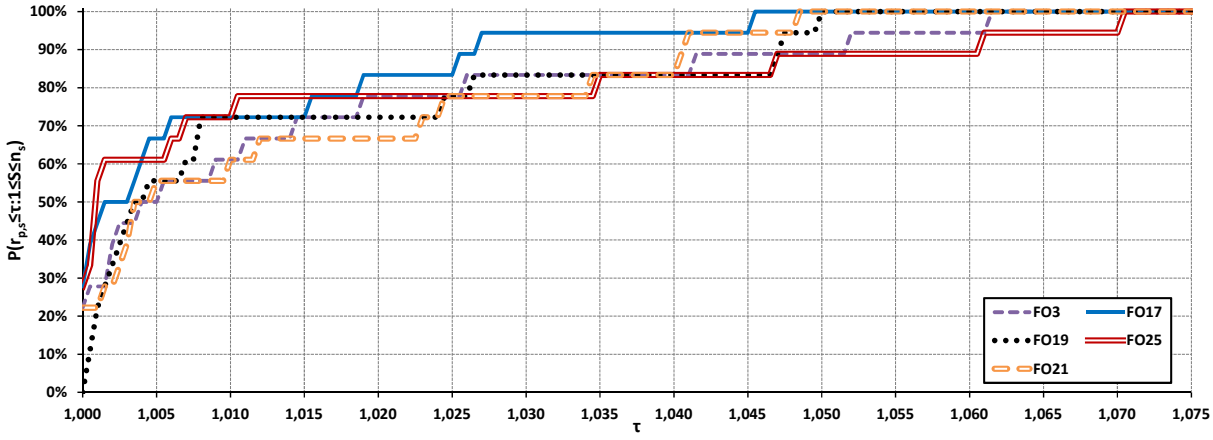

Figure 5: Graph of performance curves (DM) of the five best sets of parameters for the TFO-HCG heuristic.

Given that the range of superiority of the FO25 variant ends at a very low value of τ, the FO17 variant was chosen as the best parameterization for the TFO-HCG heuristic. Partitioning by periods and machines is composed of partitions with a size of eight sub-periods per machine, and there is an overlap of four of these sub-periods. Partitioning by items and machines is done with two grammages and two machines in each partition with an grammage overlapping between subsequent partitions. Cube-shaped partitions are constructed with sixteen sub-periods and four grammages in size. The overlaps between neighboring partitions, in this case, are eight sub-periods and/or two grammages (Table 3).

The average computational times of TFO-HCG with the different sets of parameters varied between 2236.93 and 3064.66 seconds. The FO17 variant had an advantage over FO25 in computational time. The mean solution time for FO17 was 2468.80 seconds, while FO25 had an average computational time of 2757.30 seconds

3.3.2. TFO-HCBC Parameters tuning

The TFO-HCBC heuristic presents slightly worse results than those found for the TFO-HCG heuristic, as can be seen in Table 5. The overall mean percentage deviation was 16.37%, just 0.1% lower than previously found. In the case of the five best variants, the results were slightly better than those previously found, about 0.28% higher in the case of the best set of parameters. The maximum dispersion between the averages of the 36 sets of parameters was also higher and reached 0.67% among the five best sets.

Table 5: TFO-HCBC variants average gap (%).

Variant	Gap	Variant	Gap	Variant	Gap	Variant	Gap
FO1	15.72	FO10	15.87	**FO19**	**15.07**	FO28	16.53
FO2	16.45	**FO11**	**15.57**	FO20	16.16	FO29	17.19
FO3	16.34	FO12	18.01	FO21	16.30	FO30	16.45
FO4	16.73	FO13	15.81	FO22	16.46	FO31	16.32
FO5	16.46	FO14	16.17	FO23	16.43	FO32	17.36
FO6	16.83	FO15	15.69	FO24	16.39	FO33	16.37
FO7	15.85	FO16	16.65	FO25	16.44	FO34	17.07
FO8	17.54	**FO17**	**15.63**	FO26	16.47	FO35	16.17
FO9	**15,51**	FO18	16.87	**FO27**	**15.55**	FO36	16.96

The graph in Figure 6 presents a behavior similar to that found in the graph in Figure 5, where one method does not completely dominate the others for any value of τ. However, there are some variants that stand out in relation to the others. They are candidates for the best variants FO19, FO09 and FO17. Each of them dominates in a different range of τ. Considering a general performance, it can be seen that among the three, FO19 is the variant that dominates the others for larger intervals of τ. In addition, it performs very close to the others when it is not the best variant for a given value of τ. The FO19 variant is also the first to reach 100% of solutions and the variant that has the most number of best solutions.

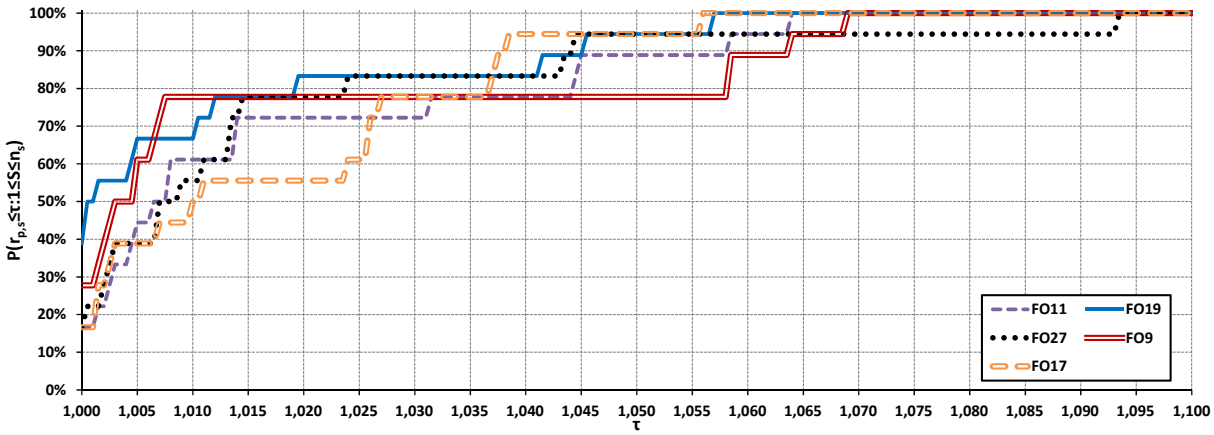

Figure 6: Graph of performance curves (DM) of the five best sets of parameters for the TFO-HCBC heuristic.

FO19 has partitioning by periods and machines of size eight sub-periods per machine and overlapping four sub-periods. Partitioning by items and machines takes into account two items by two machines in each partition, with one item shared by subsequent partitions (overlap). Finally, the cube-shaped partitioning is composed of twenty-four sub-periods and four items, sharing up to twelve sub-periods and two items with neighboring partitions (Table 3).

The average computational times of the TFO-HCBC method ranged from 2322.17 to 3159.02 seconds, with the FO19 variant having an average of 2478.92. This value was among the top five computational times of the 36 variants. The difference in time with the TFO-HCG method may be due to the need for more time for the constructive heuristic, especially in the case of larger samples.

3.3.3. VFO-HCG Parameters tuning

The results for the VFO-HCG heuristic, presented in Table 6, demonstrate a visible improvement over the results of the TFO-HCG method. These methods have the same starting solution, but differ in the way

fix-and-optimize has been implemented. The average percent deviation of all parameter sets was 14.41% versus 16.27% for the traditional implementation, a difference of 1.86%. The variation found among the five best parameter tuning of VFO-HCG was 0.03% in mean deviation and the best variant was FO35, which reached mean percentage deviation equal to 14.15%.

Table 6: TFO-HCBC variants average gap (%).

Variant	Gap	Variant	Gap	Variant	Gap	Variant	Gap
FO1	14.34	**FO10**	**14.16**	FO19	14.37	FO28	14.65
FO2	14.35	**FO11**	**14.16**	FO20	14.58	FO29	14.32
FO3	14.28	FO12	14.33	FO21	14.60	FO30	14.52
FO4	14.36	FO13	14.62	FO22	14.55	FO31	14.32
FO5	14.19	FO14	14.55	FO23	14.65	FO32	14.56
FO6	14.72	FO15	14.68	FO24	14.40	**FO33**	**14.18**
FO7	14.37	FO16	14.53	FO25	14.20	FO34	14.29
FO8	14.32	FO17	14.43	FO26	14.34	**FO35**	**14.15**
FO9	14.35	FO18	14.48	**FO27**	**14.17**	FO36	14.61

The small variation in the average gap can also be seen in the graph of performance curves in Figure 7. In this chart, τ ranges from 1 to 1.045, that is, there is no solution of these five variants that is 4.5% worse than the best solution for the same instance. Again, two variants stand out and dominate part of the chart. The FO11 variant dominates the beginning of the chart from $\tau > 1$ until close to $\tau = 1.011$, with losses in some points. After that, there is a tie with the variant FO10 up to $\tau = 1.013$, which dominates the rest of the chart. The number of best solutions of the two variants are both equivalent to one third of the solutions (6 best solutions in 18 instances). The performance of the FO11 variant was bad only for one of the intances of the parameter tuning set, the larger one. As large instances are of practical relevance (they are closer to reality), and because they have a similar performance in the other runs, the FO10 variant was chosen as the best option for fixing-and-optimizing VFO-HCG.

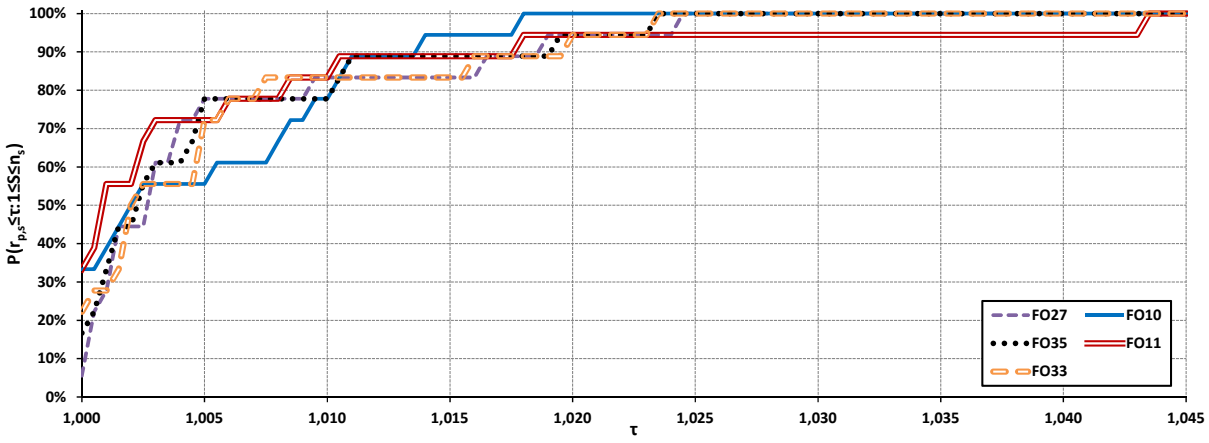

Figure 7: Graph of performance curves (DM) of the five best sets of parameters for the VFO-HCG heuristic.

In addition to improving the results regarding the average gap (Table 6), the VFO-HCG method obtained computational times 26% lower than those of the TFO-HCG heuristic. The variants achieved average computational times of 1056.88 to 2264.91 seconds and FO10 got on average 1987.13 seconds. This result is not among the best when compared to other variants, but it is 481 seconds better than the best TFO-HCG result.

The parameter tuning of the FO10 is partitioned by periods and machines. The constructed partitions

are four sub-periods in size across two machines, with two sub-periods overlapping between neighboring partitions. In partitioning by items and machines, four items per machine size were used in the partitions. The defined overlap was two items between neighboring partitions. The cube-shaped partitions have sixteen sub-periods by eight items. In addition, the overlap considered was eight sub-periods and four items (according to Table 3).

3.3.4. VFO-HCBC Parameters tuning

Table 7 presents the average gap obtained by the VFO-HCBC method with each set of parameters. The overall average result was 14.66%, 1.71% higher than the TFO-HCBC combination. The deviation of the five best variants was 0.19%, a value lower than that found in the TFO-HCG and TFO-HCBC methods.

Table 7: TFO-HCBC variants average gap (%).

Variant	Gap	Variant	Gap	Variant	Gap	Variant	Gap
FO1	**14.46**	**FO10**	**14.47**	FO19	14.74	FO28	14.78
FO2	14.65	FO11	14.49	**FO20**	**14.28**	FO29	14.79
FO3	14.68	FO12	14.70	FO21	14.50	FO30	15,09
FO4	14.53	FO13	14.89	**FO22**	**14.41**	FO31	14.71
FO5	**14.42**	FO14	14.71	FO23	14.54	FO32	14.80
FO6	14.52	FO15	14.63	FO24	14.61	FO33	14.75
FO7	14.91	FO16	14.69	FO25	14.66	FO34	14.65
FO8	14.86	FO17	14.51	FO26	14.83	FO35	14.50
FO9	14.81	FO18	14.74	FO27	14.87	FO36	14.56

Figure 8 have similar behaviors to the other graphs in the parameter tuning process of the fix-and-optimize heuristics. The FO20 and FO22 variants dominated the other parameter sets and consolidated as the best. FO22 behaves better for $\tau < 1.009$ and FO20 consolidates as the best from this point on. In this case, we have a competition between the option with the highest number of best solutions (FO22) and the set of parameters that reaches 100% with the lowest τ and the lowest average gap (FO20). As a value of $\tau = 1.009$ is equivalent to indicating a loss of 0.9%, we opted for the FO20 variant, which has greater robustness and average quality.

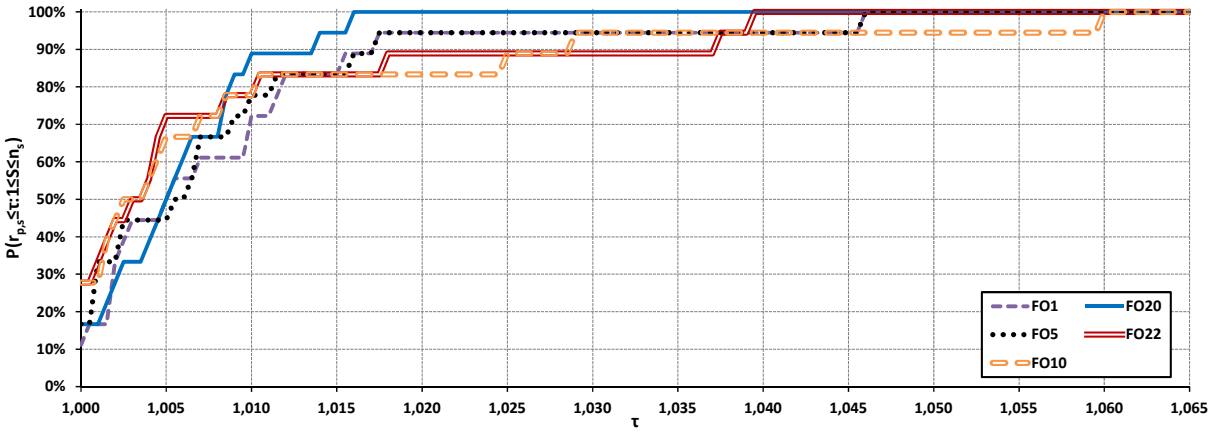

Figure 8: Graph of performance curves (DM) of the five best sets of parameters for the VFO-HCBC heuristic.

The average computational time of the VFO-HCBC method varied between 1207.67 and 2358.89 seconds and the FO20 variant had an average value of 2045.28 seconds. The improvement in computational time with respect to TFO-HCBC was similar to that found in the case of the VFO-HCG method with respect to

TFO-HCG. This indicates that the improvement in the method occurred regardless of the initial solution. The best quality partitioning, on the other hand, seems to be influenced by the initial solution and the way the method is developed, as it varied according to the combination of methods tested.

The FO20 variant features partitioning by periods and machines with eight sub-periods and one machine per partition, where four sub-periods are shared between subsequent partitions. Partitioning by items and machine was used, containing two items and two machines per partition and overlapping one item with each neighbor. The square partitions are size twenty-four sub-periods and size eight items. The overlap between neighboring partitions is twelve sub-periods and four items, according to Table 3.

References

Dolan, Elizabeth D, & Moré, Jorge J. 2002. Benchmarking optimization software with performance profiles. Mathematical programming, **91**(2), 201–213.