Supplementary Material for: A landscape-based analysis of Fixed Temperature and Simulated Annealing

Alberto Franzin[a], Thomas Stützle[a]

[a]*IRIDIA, Université Libre de Bruxelles (ULB), Belgium*

1. Quadratic Assignment Problem

1.1. Results on the random instances

In Figure 1 we report the results for random instances of size 60, 80, 100; in each plot there are the results for FTA, SA for four tuning budgets 500, 1000, 2000, 5000, plus the configuration from [1], here called `All`, as reference. In Figure 2 the same results are separated by tuning budget, and for each algorithm we have grouped the results by instance size.

A numerical comparison is given in Table 1. In most cases (11 out of 12), for the same instance size and tuning budget, FTA slightly outperforms SA; only in one case with the lowest budget the difference is not statistically significant (with a p-value of 0.2839).

For the FTA, as the budget increases, the results for a given instance size improve (though less strongly for instance size 100, it takes more budget to do so). For a budget of 500 the results worsen regularly as the instance size increases, but for higher budgets the results for sizes 60 and 80 are not significantly different; the average difference $X60 - X80$ is 0.0047, 0.0037, -0.0195 with p-values of 0.8696, 0.9308, 0.2351 respectively, for budgets of 1K, 2K, 5K. For instance size 100, the results are instead much worse.

For the SA, the results improve noticeably along with the budget for size 60, but much less so for instance sizes 80 and 100 (p-values around 0.05 between consecutive boxplots). For the same experiment budget, instead, SA results are proportional to the instance size (except for a budget of 500).

Email addresses: `afranzin@ulb.ac.be` (Alberto Franzin), `stuetzle@ulb.ac.be` (Thomas Stützle)

Table 1: Average difference (FTA - SA) for each instance size and tuning. A negative value means that FTA outperforms SA. Grey background means that the results are **not** statistically significant with a p-value greater than 0.05 according to a paired T-test; a green background means the p-value is smaller than 10^{-4}.

	500	**1K**	**2K**	**5K**
60	-0.1004	-0.0270	-0.0235	-0.0659
80	0.0260	-0.0387	-0.0625	-0.0769
100	-0.0151	-0.0206	-0.0766	-0.0543

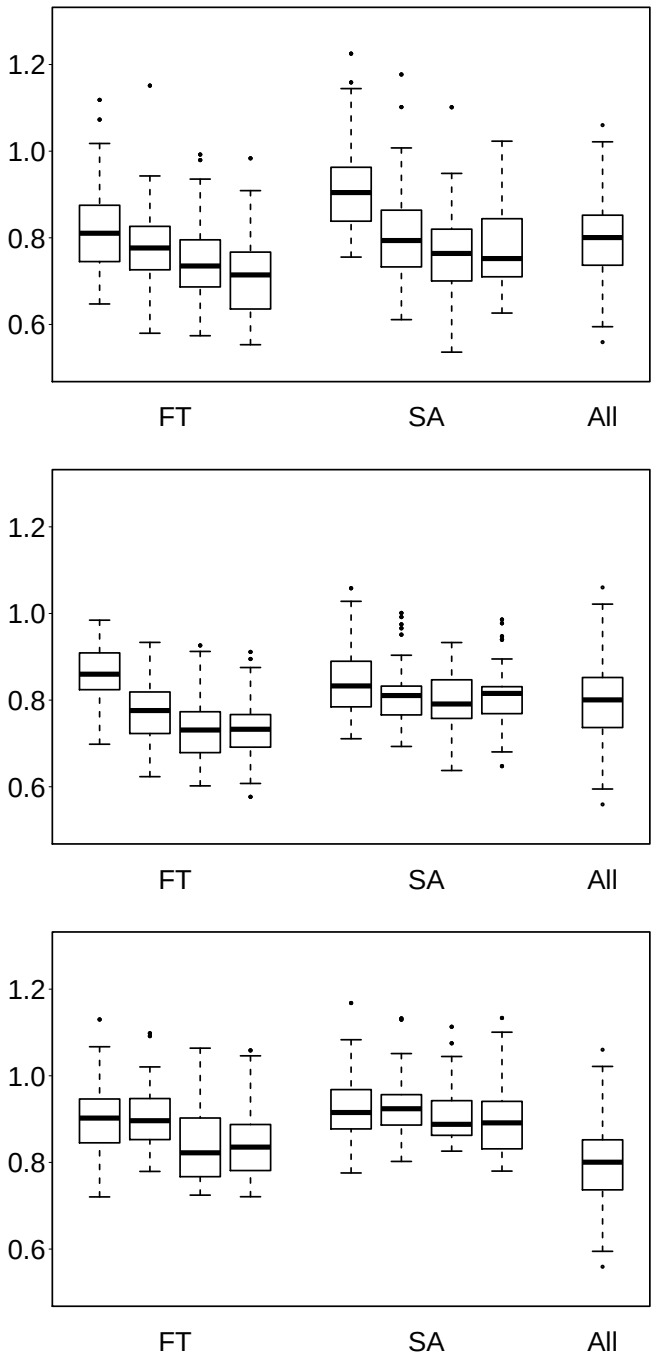

Figure 1: Average Relative Percentage Deviation (ARPD) for random instances on, respectively, instances of size 60, 80, 100. For each algorithm, the boxplots report the results obtained with tunings with budget, respectively, 500, 1K, 2K, 5K.

Summary of the composition of the FT algorithm. Looking at the algorithms found with the highest budget, there seem to be a sweet spot for which the best results can be obtained, that is, an "asymptotic" configuration composed by (i) an initial temperature value obtained with an initial temperature scheme proportional (by a factor of around 0.22) to the average gap between consecutive solutions in a random walk, and (ii) a sequential exploration. The performance of this algorithm (with a factor of 0.2196 to

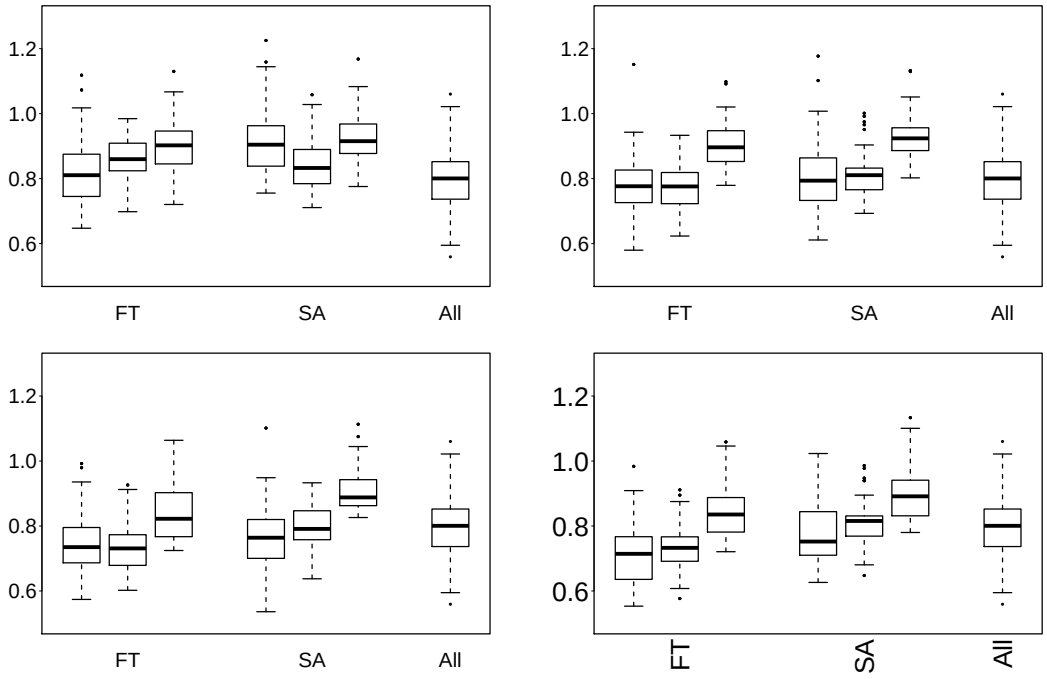

Figure 2: Average Relative Percentage Deviation (ARPD) for random instances for tunings of budget, respectively top to bottom and left to right, 500, 1K, 2K, 5K. For each algorithm, the boxplots report the results obtained on instances of size 60, 80, 100.

match one of the configurations already obtained and be able to crosscheck the results) are reported in Figure 3 labeled as `fix`, and is even slightly better than the previously found configurations, event when tuned with a budget of 5000 experiments on separated instance sizes.

Thus, it looks like the initial temperature value is related to the instance size, by means of the relative difference between solutions. With a scaling coefficient of 0.22, the acceptance probability of an average worsening move (that is, worse by the average gap found during the initial random walk) is around 1%.

These results are consistent across the three instance sizes and, as the tuning budget increases, the best algorithms resemble more and more the configuration here described.

The results for FTA size 100, which are not as good as the other instance sizes also with a tuning budget 5000 of experiments, reflect a lower number of final configurations that match the best one. The results of `fix` for size 100, instead, simply reflect the lower number of moves that can be evaluated in the given time. The first plot of Figure 4 shows the performance of `fix` over different runtimes, while the third one shows the results when considering the same number of moves evaluated (respectively, 2, 20, 200, 2000 million moves). In the first plot the results for different instance sizes are clearly much more similar than in the other ones, though there is still a slight statistical difference observed between different experiments (except for sizes 80 and 100 for two million moves).

In the second plot of Figure 4 we see that the perfect dependence of the results of the `fix` configuration on the relative difference between solutions and not on the specific instance values is confirmed by the results obtained when rescaling all the instance values. For the same random seeds, the search progression is exactly the same, thus the solutions discovered are the same. The only occasional difference in the results is due to the

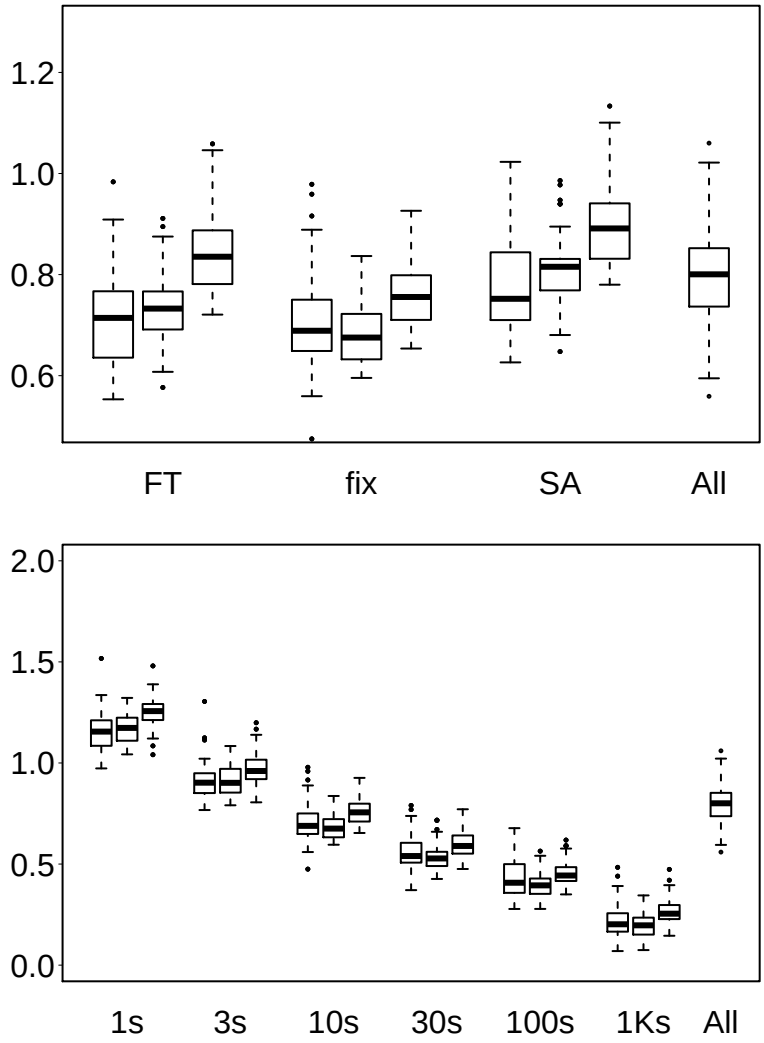

Figure 3: Comparison in terms of Average Relative Percentage Deviation (ARPD) for random instances of the inferred best configuration (`fix`) against FTA and SA tuned with a budget of 5000 experiments separately on instances of size 60, 80, 100. In the second plot, the `fix` configuration is evaluated over 1, 3, 10, 30, 100, 1000 seconds.

intrinsic stochasticity of the computational environment, that is, sometimes a different number of solutions can be evaluated in the given time, and a new improving solution recorded at the end of a run is missed in another one simply because that solution was not reached by the search.

1.1.1. Random exploration

We ran an additional tuning, where the neighbourhood exploration scheme is now fixed as the random exploration, and the coefficient of the initial temperature is the only tunable parameter. The tuning setup is: 15 tunings, 1000 experiments per tuning, a $(10, 2)$ statistical test policy for each race, to ensure enough instances of different size are seen before each pruning. The tunings converge to a coefficient of, on average, 0.1799, with a very low standard deviation (0.0039). This corresponds to an acceptance probability of 0.38% for an average random solution, using the Metropolis formula. This indicates that the random exploration has a more diversifying behaviour than the sequential one,

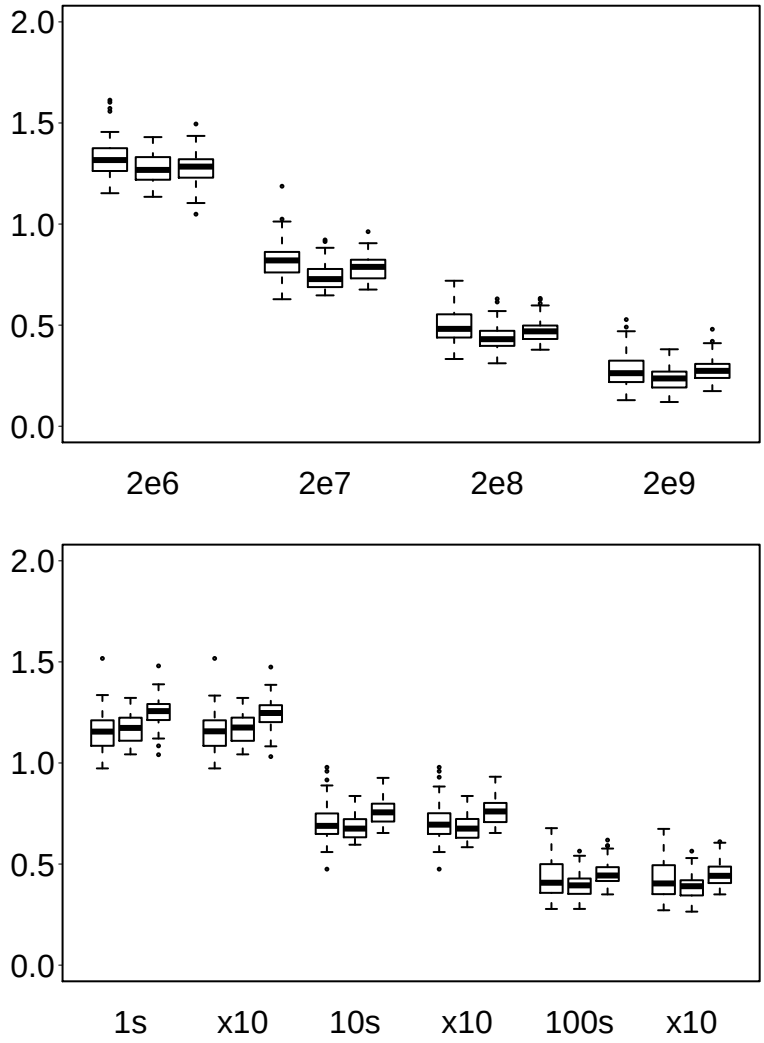

Figure 4: Comparison in terms of Average Relative Percentage Deviation (ARPD) for random instances of the inferred best configuration (`fix`) evaluated over the same number of moves (2×10^6, 2×10^7, 2×10^8, 2×10^9, first plot), and on $\times$10-rescaled instances (side-by-side original and rescaled, for 1, 10, 100 seconds, second plot).

and the acceptance has to be stricter to compensate.

The results are shown in Figure 5. We see that even with the best configuration the results are not as good as with the sequential exploration, and, for the ten seconds runtime, worse than the ones obtained by the fully-tuned SAs.

Summary of the composition of the SA algorithm. The higher number of choices makes it more difficult to reach a "consensus" configuration; furthermore, as observed in the previous paper, there is a number of different configurations that obtain results of comparable quality, in the given tuning scenario. Nonetheless, there still are some things we can say. The first observation is that the "best-of-k" exploration (or its first improvement variant) with k covering roughly 1/4 of the neighbourhood size appears quite often, consistently across instance sizes, confirming the observations of the previous section. The second one, strengthened by the results of the previous paper, is that on these QAP random instances the SA algorithms discovered do not outperform FT.

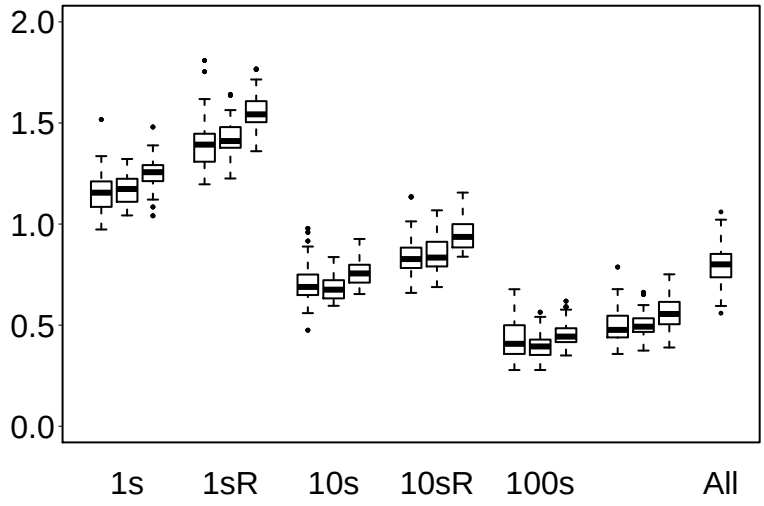

Figure 5: Comparison in terms of Average Relative Percentage Deviation (ARPD) for random instances of the average best configuration using the sequential exploration (see main paper for details), and the best configurations using random exploration for 1, 10 and 100 seconds, side-by-side.

1.2. Results on the structured instances

The results that can be obtained by FTA on the structured instances are very far from the results of a SA algorithm, for each of the instance sizes and tuning budgets considered. The results are reported in Figures 6 and 7 divided, respectively, by instance size and tuning budget.

We clearly see that already for a budget of 500 experiments the SA configurations obtains results that FTA cannot reach even with a budget of 5000. For a budget of 5000, the SAs obtain results comparable with the fully-tuned SAs of [1].

This is surely caused by the structure of the instance: if there is an "easy" valley in the fitness landscape, too much diversification will hamper the search convergence; however, the structure of the slope makes it difficult, probably impossible, to find generic settings that work fine for both the initial and the latter stages of the search. In this regard, SA can tailor a cycle of exploration-exploitation transitions that better fit the fitness landscape.

As the results of FTA are very poor, we do not perform additional experiments, since we do not expect significant variations in the results.

Summary of the composition of the SA algorithms. Again, the composition of the algorithms is very diverse, but in this case the results are consistently good.

The options that appear more often are (i) the random exploration, and (ii) the geometric cooling scheme, and (iii) initial temperature schemes based on a preliminary random walk (though, by looking at the temperature values of some algorithms, the values vary wildly).

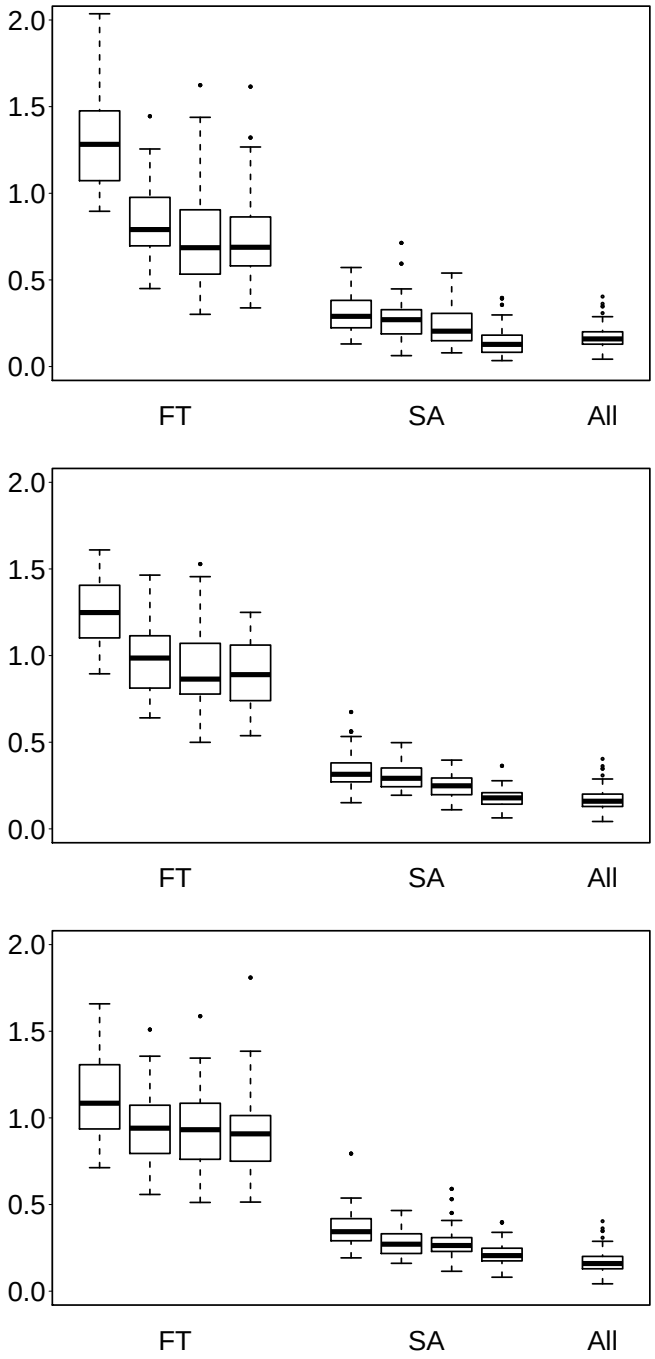

Figure 6: Average Relative Percentage Deviation (ARPD) for structured instances on, respectively, instances of size 60, 80, 100. For each algorithm, the boxplots report the results obtained with tunings with budget, respectively, 500, 1K, 2K, 5K.

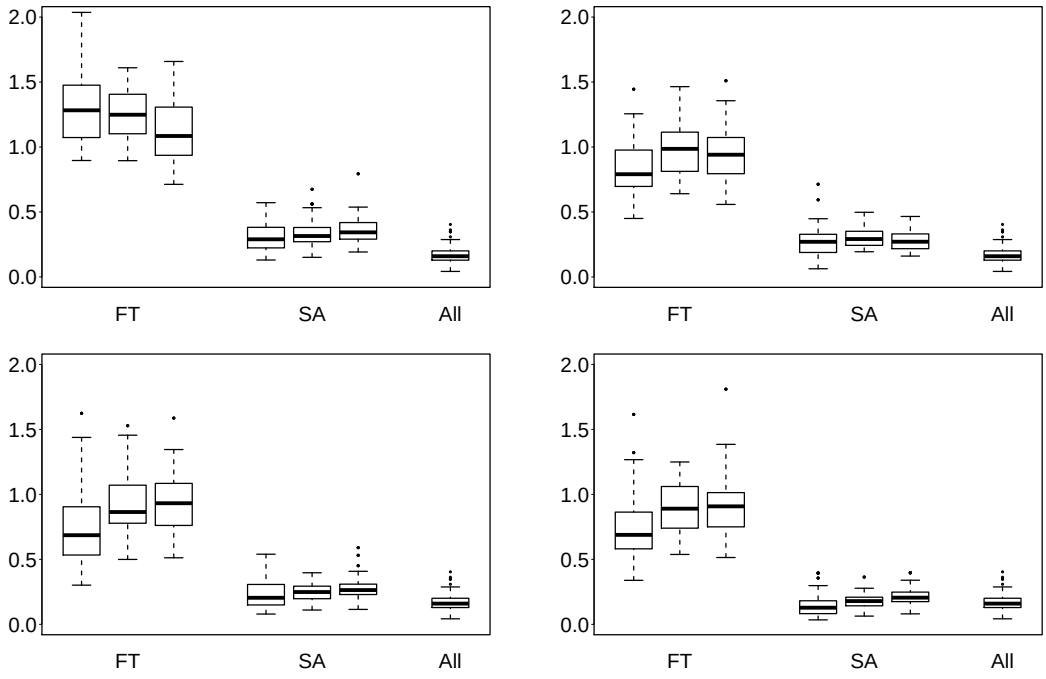

Figure 7: Average Relative Percentage Deviation (ARPD) for structured instances for tunings of budget, respectively top to bottom and left to right, 500, 1K, 2K, 5K. For each algorithm, the boxplots report the results obtained on instances of size 60, 80, 100.

2. Permutation Flowshop Problem

In Figures 8 and 9 we show the results obtained by our set of algorithms on the Taillard benchmark, for the Makespan and Total Completion Time objectives respectively, in terms of relative percentage deviation (RPD) with respect to the optimal or best known solutions. In each scenario, we report the results for FT, FT.P, SA, each one in four variants, in the following order: (i) initial solution chosen at random and exchange neighbourhood, (ii) initial solution chosen at random and insertion neighbourhood, (iii) initial solution computed by the NEH heuristic and exchange neighbourhood, and (iv) initial solution computed by the NEH heuristic and insertion neighbourhood. For a comparison, we include also the RPD of the fully-designed SA algorithms of SA PAPER.

The range of results is relatively wide compared to the results on the QAP, because the test set contains instances of very different sizes, and different jobs/machine ratio. Moreover, we use a training set composed of instances with a different number of jobs and machines, we present the results on the overall benchmark (120 instances, first plot in Figures 8 and 9), on the subclasses of instances whose number of jobs is included in the training set (80 instances, second plot), and the subclasses of instances whose number of jobs and machines is included in the training set (30 instances, third plot). The full breakdown of subclasses of instances is reported in the supplementary material.

On the overall Taillard benchmark under the Makespan objective, FT obtains good results, close to those obtained by the two other algorithms (when using the same neighbourhood). The main contribution in difference of the results comes from the thirty smallest instances (20 jobs, smaller than any instance in the training set), on which the ILS and SA have better results, and with the relative performance between algorithms worsening as the number of machines increases. On the other instances, a paired Wilconox test confirms that FT results are not statistically different from the ones of SA (p-value of 0.2368, checked for random initial solution and insertion neighbourhood, the combination with the best results), while FTP is slightly, but statistically better than the other two algorithms (p-value of 5×10^{-8}). On the third group of instances (test set with same number of jobs and machines of the training set) the results are even closer, and narrower in terms of variance. SA apparently adapts better to the "new" scenarios, thanks to its higher flexibility than FT. In general, the better performance is obtained for high jobs/machines ratio, despite the fact that those instances have a number of machines lower than what we have in our training set. We observe also a general higher consistence of the results in each subclass of instances, except for instances of size 50×10.

FTP and SA are not significantly different for the same initial solution and neighbourhood in three out of the four cases, and barely different in the case of NEH and exchange neighbourhood (the four p-values are 0.1118, 0.3345, 0.0292, 0.3857, computed with a paired Wilcoxon test on the whole test set).

Interestingly, the initial solution generation does not have a significant effect on the performance of the algorithms overall, at least in the settings used in our experiments (it may be more relevant in case of extremely short running times). For the same algorithm and neighbourhood, only FT and FTP with exchange neighbourhood obtain significantly different results on the whole benchmark, with p-values respectively of 0.9×10^{-9} and 0.0012 – all the other ones being above 0.1. The main difference is done by the neighbourhood, with the insertion neighbourhood clearly outperforming the exchange neighbourhood. The effect of the latter is in fact to generate a more rugged landscape, more difficult to navigate and in which the algorithms usually get stuck in a poor local optimum in relatively few moves.

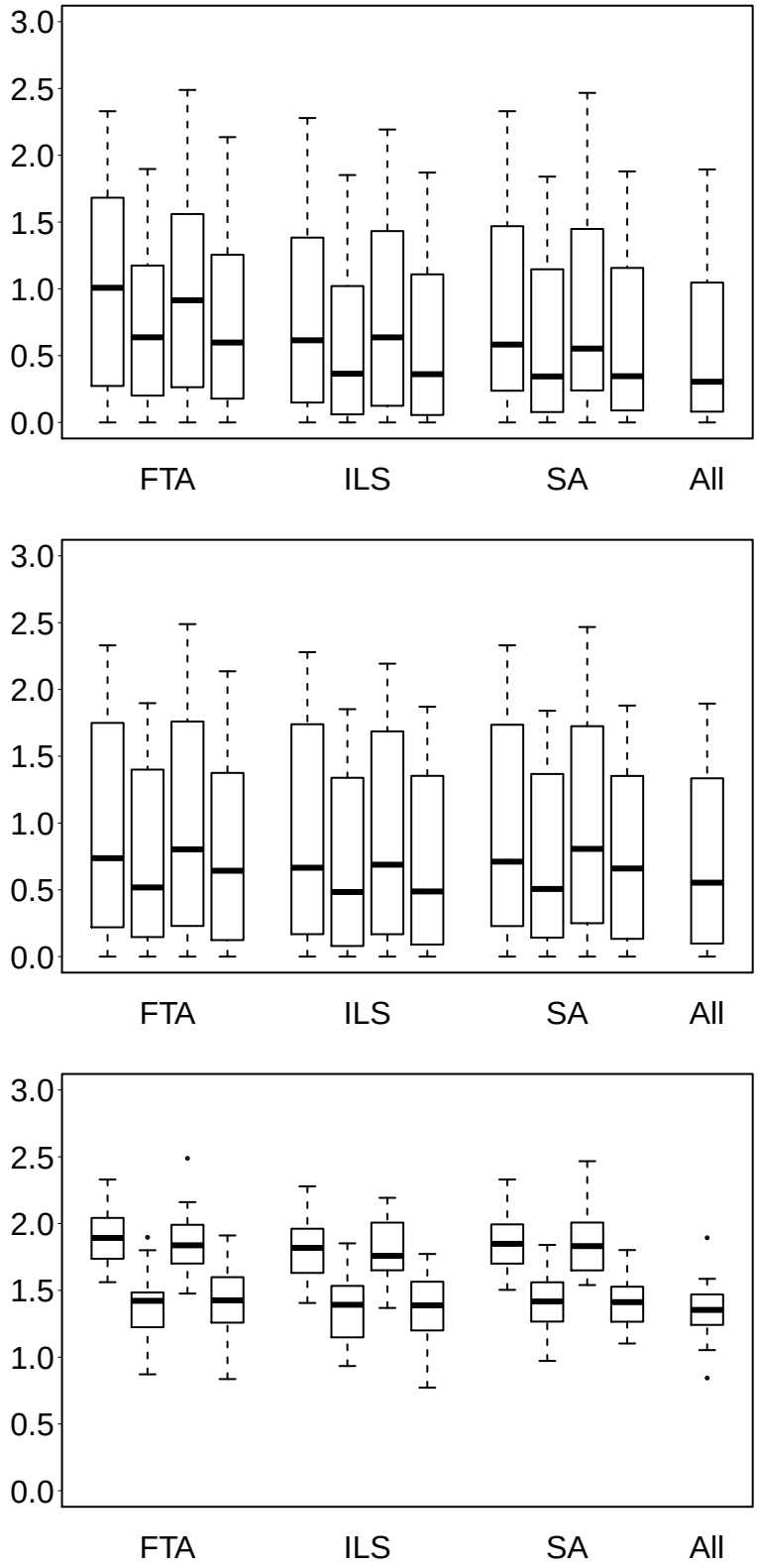

Figure 8: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Makespan objective on the Taillard benchmark: (i) all instances, (ii) instances with a number of jobs included in the training set, (iii) instances with a number of jobs included in the training set. Details about the neighbourhoods and initial solution schemes are in the text.

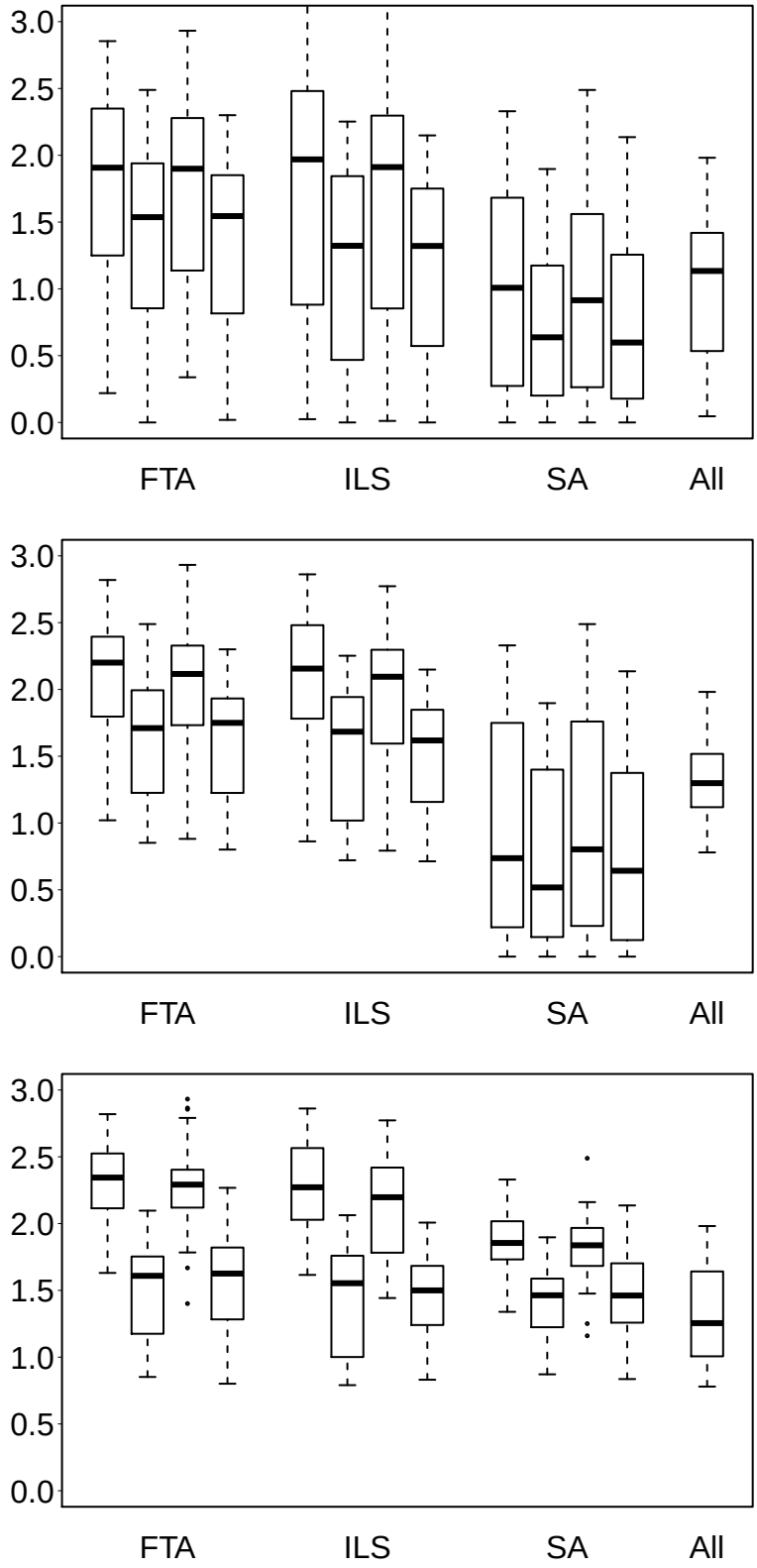

Figure 9: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Total Completion Time objective on the Taillard benchmark: (i) all instances, (ii) instances with a number of jobs included in the training set, (iii) instances with a number of jobs included in the training set. Details about the neighbourhoods and initial solution schemes are in the text.

When using the TCT objective FT and FTP do not obtain, in general, results close to those obtained by the SA. FTP improves over FT, but enough to match the results of SA. On the other hand, the performance of FTP with the exchange neighbourhood is less consistent than when using the insertion neighbourhood, resulting, in some instances, in final solutions worse than those found by the corresponding FT. We explain this with the fact that in our setup the perturbation consists in a fixed number of moves, and this does not adapt well to the wide range of landscapes generated by instances of different characteristics; therefore, whereas a certain perturbation strength is good on certain instances, it become a hindrance on very different instances. This can be observed in particular when focusing on the instances whose number of jobs or machines is covered by the training set. In the first case, FT and FTP are closer in performance, though still statistically different according to a paired Wilcoxon test; however, in this case the p-values of a non-paired Wilcoxon test are 0.0819 and 0.0729 between FT and FTP with insertion neighbourhood, for the two initial solutions schemes. In the second case, instead, while using the exchange neighbourhood does not allow FT and FTP to match the results of SA, with the insertion neighbourhood the results are almost equivalent.

We also observe two interesting trends: as the ratio jobs/machines decreases, the insertion neighbourhood increases its advantage over the exchange one, and (with it) the relative performance difference between the algorihms decreases.

In Figure 10 we show the results obtained on the Watson benchmark; given the similarity of results, we include only FT and SA, in the same four variants as in the Taillard case.

This is an easier benchmark than the Taillard one, with an extremely flat and smooth solution landscape, where a randomly generated permutation already is expected to be less than 4% away (and on some instances, less than 0.5% away) from the optimal or best known solution, and where a simple iterated improvement is sufficient to find the optimal solution on some instances. Furthermore, the test set comes from the same instance distributions of the training set. It is therefore not surprising that all the algorithms are, for both objectives, around or close to the optimal or best known solutions. On several instances, new best solutions have been found. The count for optimal or best solutions found or improved is reported in Table 2 for the Makespan objective, and in Table 3 for the TCT objective.

On the Makespan objective, despite the similarity of the results, there is a statistically significant difference between the four FT algorithms, with p-values all below 10^{-5}. The SA algorithms are all statistically significantly better than the FT ones, but not among them: SA.NEX and SA.NIN are not different (p-value of 0.3105), and so are SA.REX and SA.NEX (0.6983) and SA.RIN and SA.NIN (0.0691).

On the TCT objective, similarly, only FT.NEX and FT.NIN (p-value of 0.2705) and SA.RIN and SA.NIN (0.07521) are not statistically different; FT.REX and FT.RIN are statistically significantly different with a p-value of 0.0256, and the other pairs all have p-values below $10^{-}6$.

In general, on the two PFSP objectives we tested, FT (and its ILS variant) can obtain results equivalent to those of SA; it is however crucial to choose the right subsidiary components (neighbourhood and neighbourhood exploration) and to tune the temperature value on an instance distribution that matches the test set. When the test instances have different characteristics than the ones used for the training, SA is instead a better choice.

Algorithm	Average RPD	# best matched(/320)	% best matched	# best improved (/320)	% best improved
FT.REX	-0.0249	166	51.875	117	36.5625
FT.RIN	-0.0277	173	54.0625	121	37.8125
FT.NEX	-0.0297	170	53.125	120	37.5
FT.NIN	-0.0313	181	56.5625	123	38.4375
SA.REX	-0.0315	173	54.0625	117	36.5625
SA.RIN	-0.0325	173	54.0625	119	37.1875
SA.NEX	-0.046	76	23.75	116	36.25
SA.NIN	-0.0464	74	23.125	119	37.1875

Table 2: Summary of results about new best solutions on the Watson benchmark for the Makespan objective.

Algorithm	Average RPD	# best matched(/320)	% best matched	# best improved (/320)	% best improved
FT.REX	0.0192	11	3.4375	38	11.875
FT.RIN	0.0235	10	3.125	18	5.625
FT.NEX	0.0188	10	3.125	45	14.0625
FT.NIN	0.0259	14	4.375	15	4.6875
SA.REX	-0.0004	15	4.6875	84	26.25
SA.RIN	-0.0031	12	3.75	94	29.375
SA.NEX	0.0003	15	4.6875	84	26.25
SA.NIN	-0.0032	14	4.375	93	29.0625

Table 3: Summary of results about new best solutions on the Watson benchmark for the TCT objective.

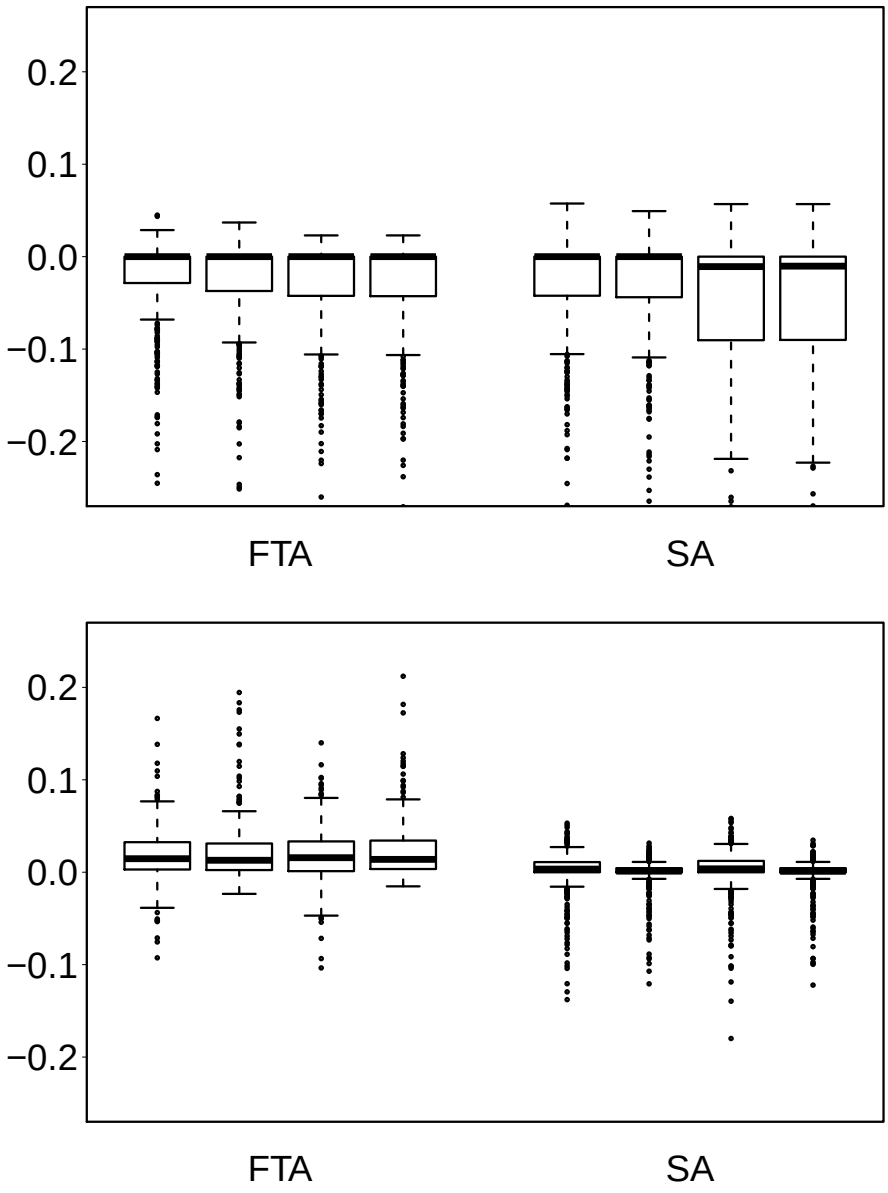

Figure 10: Comparison of the results obtained by FTA and SA, under the Makespan and Total Completion Time objectives on the Watson benchmark. Details about the neighbourhoods and initial solution schemes are in the text.

2.1. Configurations obtained

2.1.1. Taillard benchmark

Makespan objective. We analyze the configurations obtained by the automatic configuration process, starting with the Makespan objecive. In general, the vast majority of the algorithms obtained by the automatic configuration use an initial temperature proportional the average difference observed between consecutive solutions in a random walk. Using the exchange neighbourhood, for FT we have the average coefficient chosen is 0.121 when starting from a random initial solution, and 0.139 when starting from the solution computed by the NEH heuristic. When introducing a perturbation, the coefficient is around 0.055 for both the initial solution strategies; the size of the perturbation is usually very small (less than three moves), with few outliers between 22 and 90.

Using the insertion neighbourhood, the scaling coefficient for the initial temperature value is instead much more regular, being 0.066 for FT, 0.068 for FTN, 0.063 for FTFT, 0.059 for FTFTN. ALso in this case, the perturbation size is very small (up to six) with some exceptions ranging from 13 to 76.

Many of the algorithms (and all the ones employing a perturbation phase) use a random selection of the next solution to evaluate; however, some use a first or best improvement in a random subsample of the neighbourhood.

The configuration of algorithms without perturbation resulted however in a wider range of choices; while the most common options are the one reported, other combinations appeared, in particular for the initial temperature value when not using the perturbation. The overall behaviour of the algorithm is, hovever, similar to the average one described.

Regarding the SA algorithms, instead, the set of options chosen for the initial temperature is much wider. Many of the algorithms feature instead a cooling coefficient in the range $0.6 - -0.9$, no temperature restart, a temperature length based on a maximum number of accepted moves, and random exploration of the neighbourhood.

Total completion time objective. In general, for the TCT objective we observe a more diverse set of configurations than in the MS objective. Using the exchange neighbourhood and starting from a random permutation as initial solution, in nine out of 15 cases the initial temperature scheme is the one based on the average gap with a scaling coefficient of 0.0659 (on average), paired with a random neighbourhood exploration. In only one case the same initial temperature scheme appears with a best improvement in a small portion of the neighbourhood, and in this case the scaling coefficient is 0.549. In the remaining five cases the neighbourhood exploration is a first or best improvement in the neighbourhood, paired to a different initial temperature scheme. Similarly, when using the NEH heuristic to compute an initial solution the random exploration gets always paired with an initial temperature based on the average gap, with an average scaling coefficient of 0.064, for a total of nine configurations; twice the same initial temperature scheme is associated to a very short best improvement in the partial neighbourhood, with scaling coefficients 0.147 and 0.528. The remaining four algorithms feature a first improvement in the partial neighbourhood, with different initial temperature schemes that yield a higher temperature than the configurations previously described.

The same trend is observed when using the insertion neighbourhood. In nine out of 15 FT algorithms that start from a random initial solution the random exploration is paired with an initial temperature based on the average gap with an average scaling coefficient of 0.076; the remaining six FT algorithms have a partial local search in the neighbourhood coupled with different initial temperature schemes yielding a higher initial temperature. When starting from NEH, 11 out of 15 FT algorithms have random exploration and average gap-based initial temperature with a scaling factor of 0.071.

When using the perturbation, all the configurations use an initial temperature based on the average gap in a random walk: using the exchange neighbourhood, the scaling coefficients starting from random and NEH-computed solutions are, respectively, 0.051 and 0.053; using the insertion neighbourhood the coefficients with random and NEH initial solution are, respectively, 0.063 and 0.058. With the exchange neighbourhood the size of the perturbation is mostly 1 and 2 (from random and NEH initial solution), while with the insertion neighbourhood the size is up to seven for both initial solution strategies; in all the four cases ther are some occasional outliers with values up to 90. Random exploration is always used with the insertion neighbourhood. It appears also in

seven (starting from a random solution) and three (starting from from NEH) cases with the insertion neighbourhood; the remaining configurations use a sequential exploration of the neighbourhood.

Simulated annealing algorithms, in all four combinations of neighbourhoods and initial solutions, are all very different; the only common characteristic is the random exploration of the neighbourhood, which appears in all the 60 configurations.

2.1.2. Watson benchmark

Makespan objective. The results for the Watson benchmark are more homogeneous, and so are the configurations resulting from the automatic tuning for the Makespan objective. The vast majority of the fixed temperature configurations have an initial temperature based on the average gap between consecutive solutions in a random walk, paired with a random exploration of the neighbourhood. Using the exchange neighbourhood the coefficients for the initial temperature values are, on average, 0.06 and 0.051 starting from a random initial solution and with the NEH heuristic respectively, while using the insertion neighbourhood the coefficients for the two cases are 0.098 and 0.093. In few cases (5 out of 60) there is a different initial temperature scheme, combined with a different neighbourhood exploration (first or best improvement in a small subset of the neighbourhood – $< 1\%$).

The SA algorithms, instead, regardless of neighbourhood and initial solution scheme, exhibit a large variety of choices and combinations; the only steady non-fixed choice is the random exploration of the neighbourhood.

Total Completion Time objective. In the TCT case, instead, the configurations are very different; the only shared feature is a partial local search in the neighbourhood as exploration criterion, with the random exploration appearing only once. This is probably the factor that controls the exploration/exploitation tradeoff precisely enough to reach the best results; in fact, very different fixed initial temperatures appears.

The SA algorithms, instead, are very similar to those of the Makespan objective: many different configurations that share only the random exploration of the neighbourhood.

2.1.3. Comments on the configurations obtained

In general we observe a regularization effect introduced by the perturbation. The values of the initial temperature of the intensification FT are more consistent, and the same applies to the choice of components. This observation is true across different neighbourhoods and initial solutions. We explain this by the fact that the separation of the intensification and diversification phases makes it possible to have each step tailored for its specific task. There is instead no noticeable impact by the use of the NEH heuristic for the initial solution, as the results are not significantly different from the results obtained starting from a random initial solution.

On the Watson benchmark, the large variety of configurations that obtain similar results is probably just testament of the easiness of the benchmark.

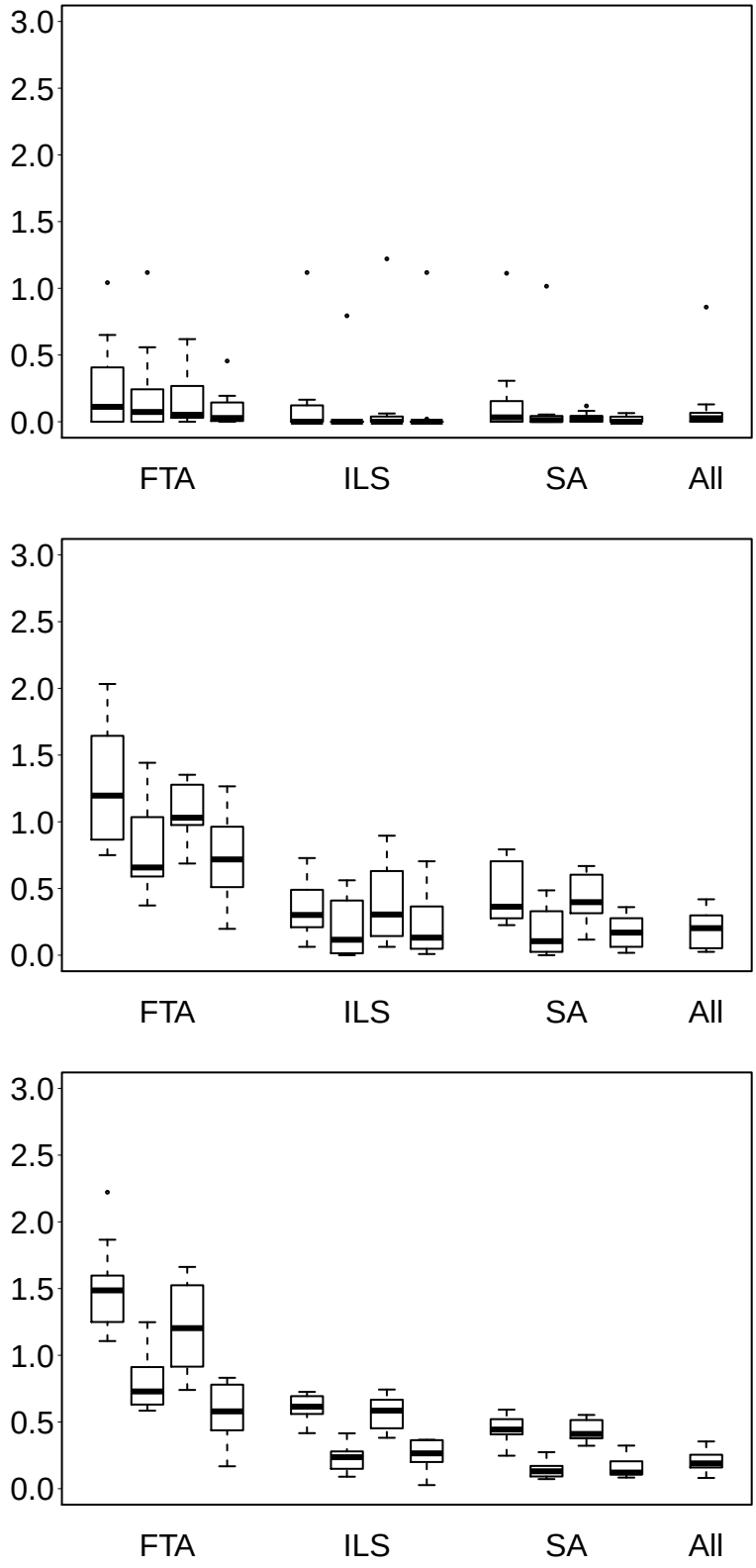

Figure 11: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Makespan objective on the Taillard instances of sizes 20×5, 20×10, 20×20.

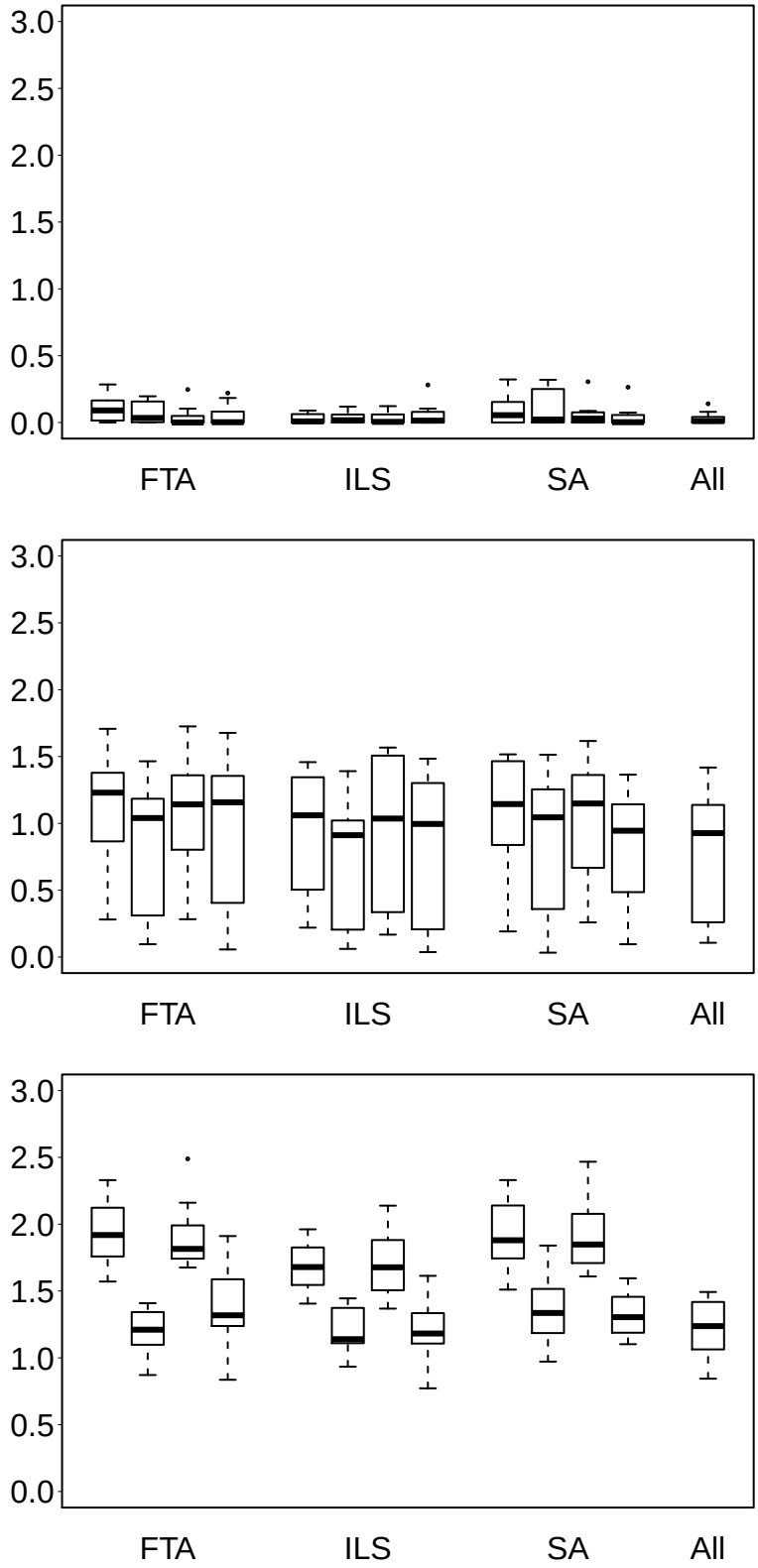

Figure 12: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Makespan objective on the Taillard instances of sizes 50×5, 50×10, 50×20.

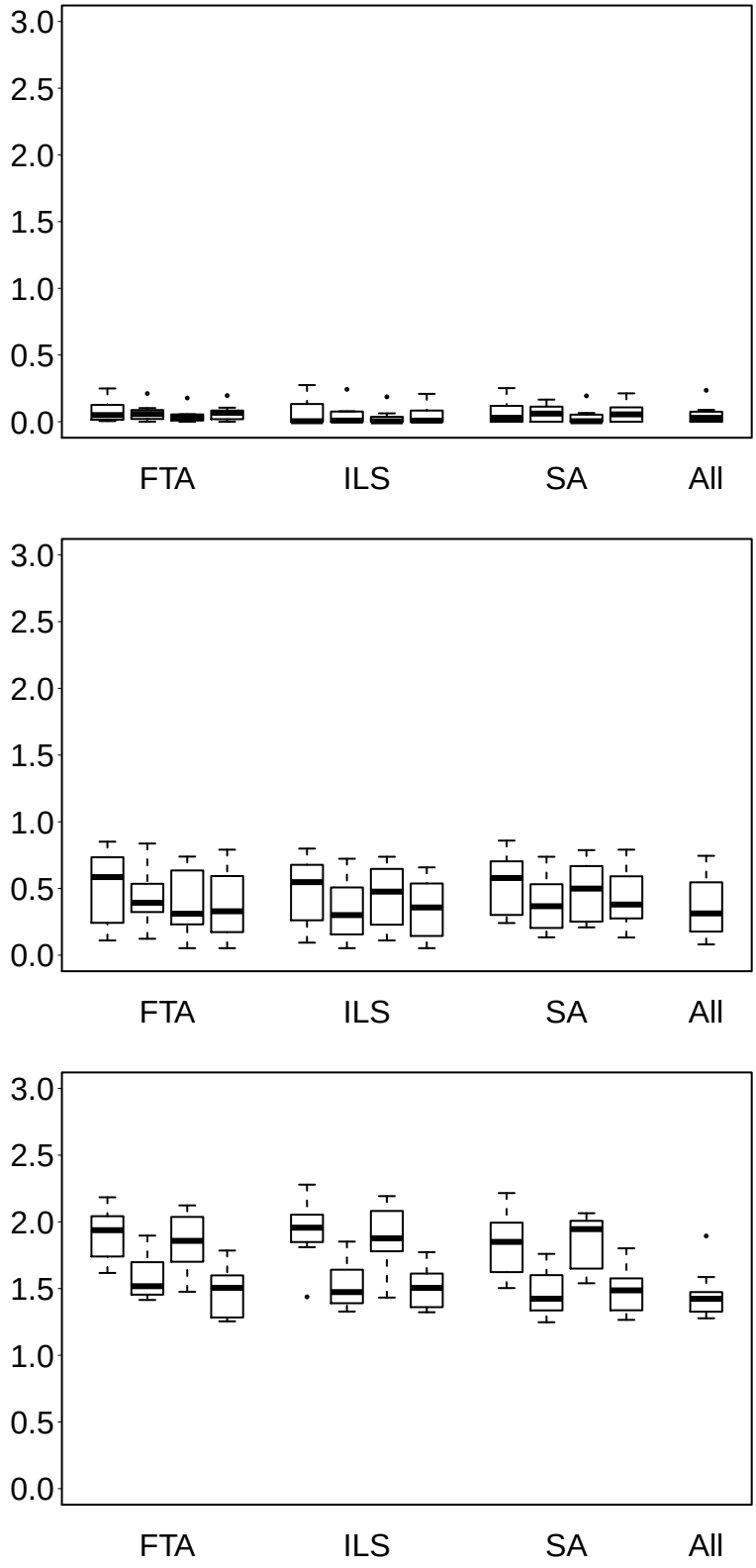

Figure 13: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Makespan objective on the Taillard instances of sizes 100×5, 100×10, 100×20.

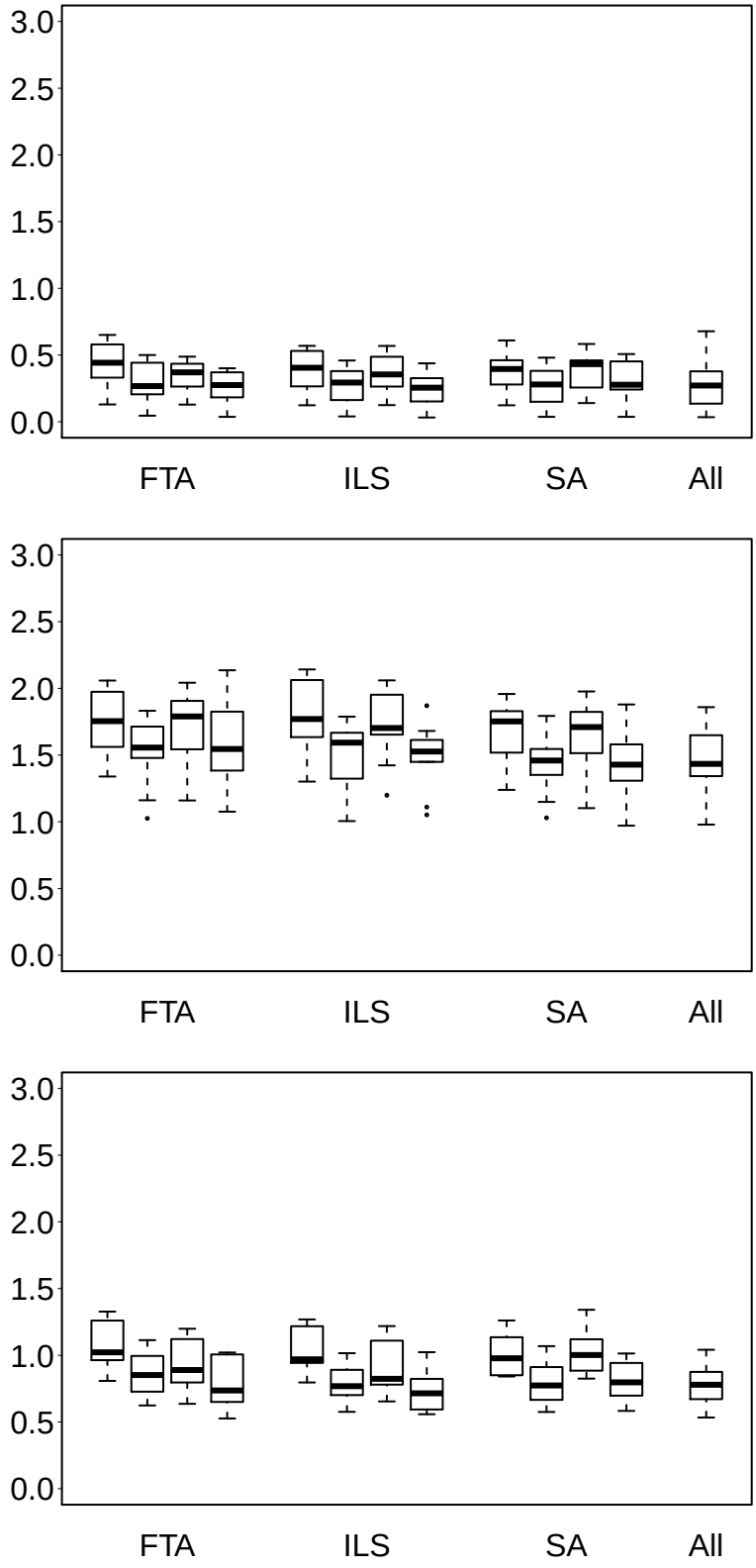

Figure 14: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Makespan objective on the Taillard instances of sizes 200×10, 200×20, 500×20.

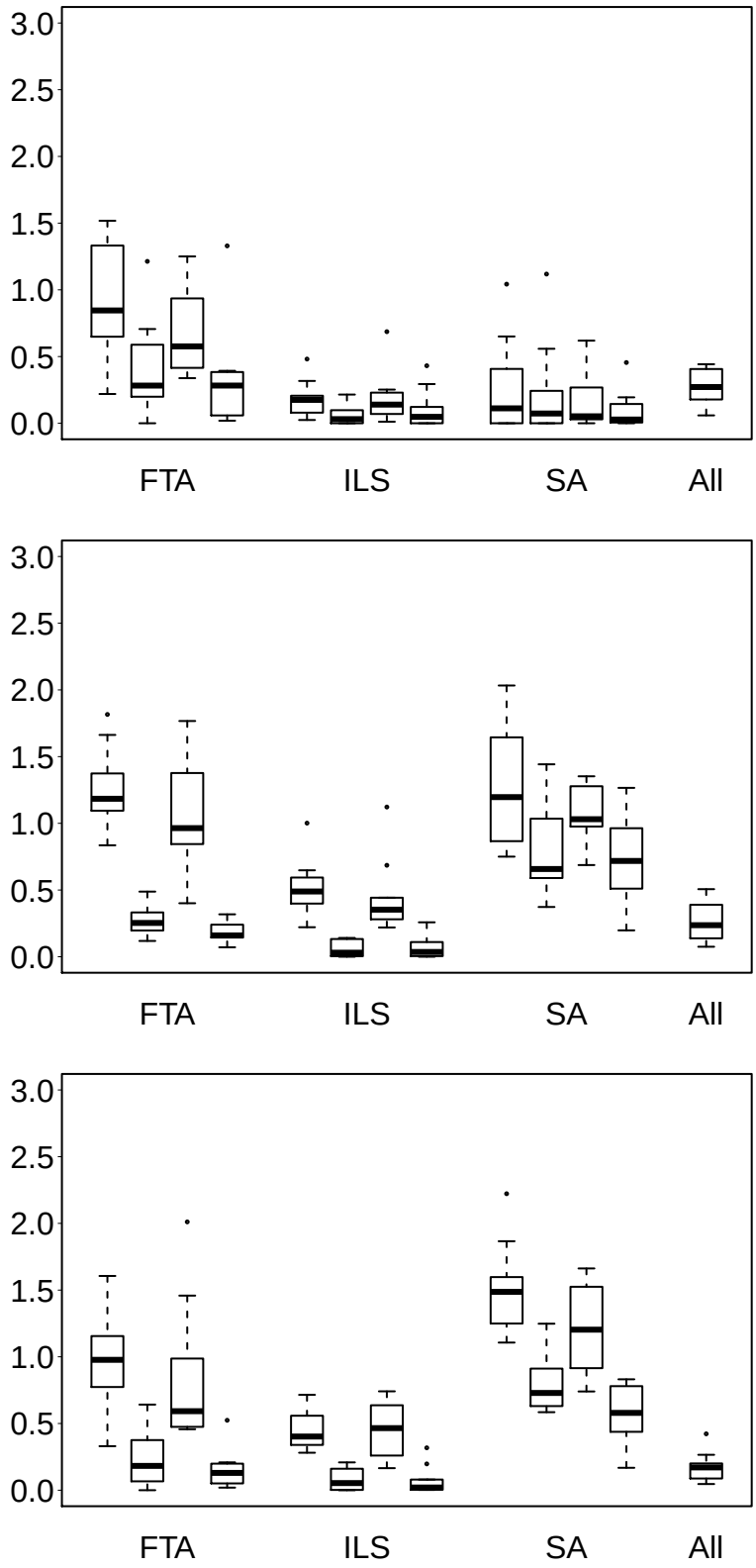

Figure 15: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Total Completion Time objective on the Taillard instances of sizes 20×5, 20×10, 20×20.

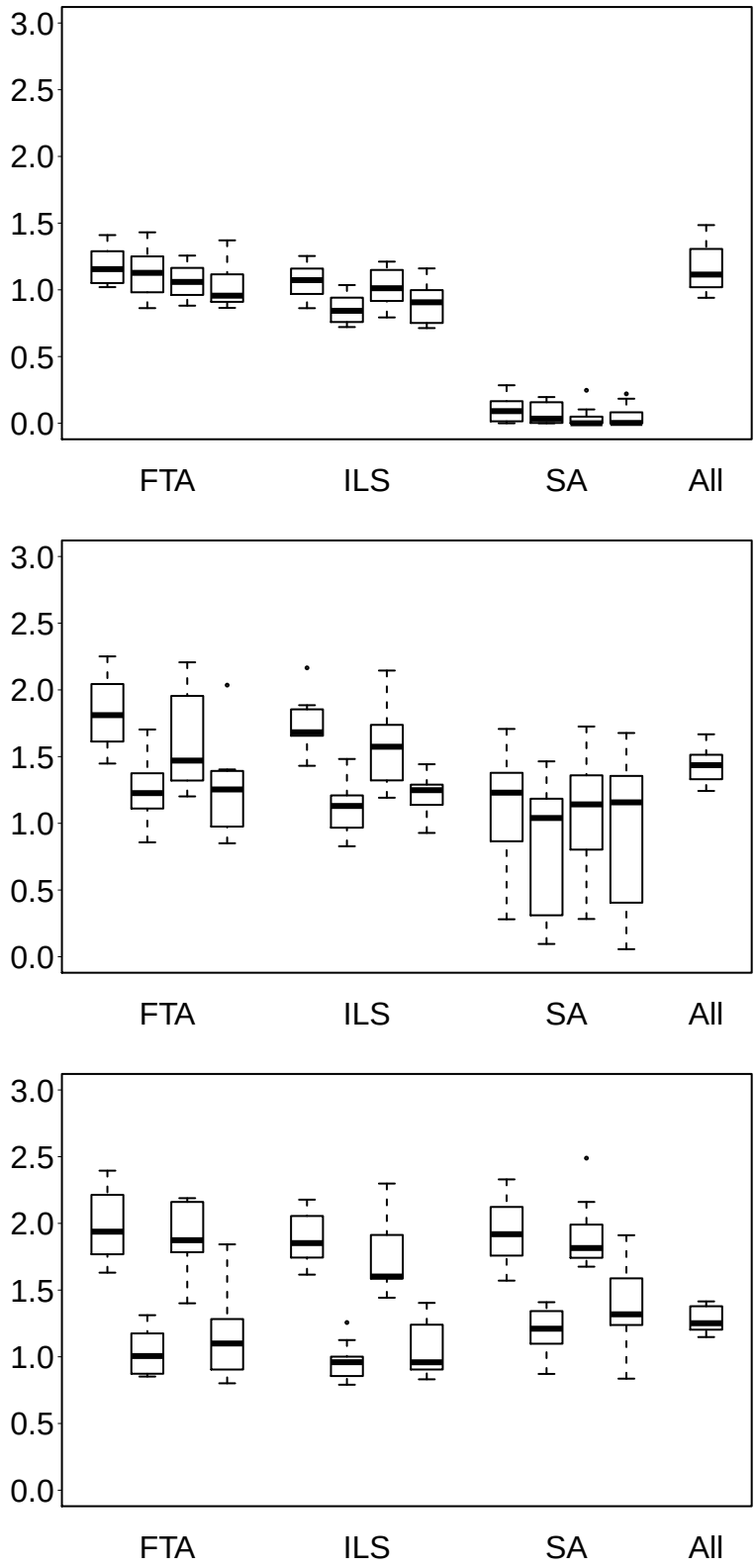

Figure 16: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Total Completion Time objective on the Taillard instances of sizes 50×5, 50×10, $\times 20$.

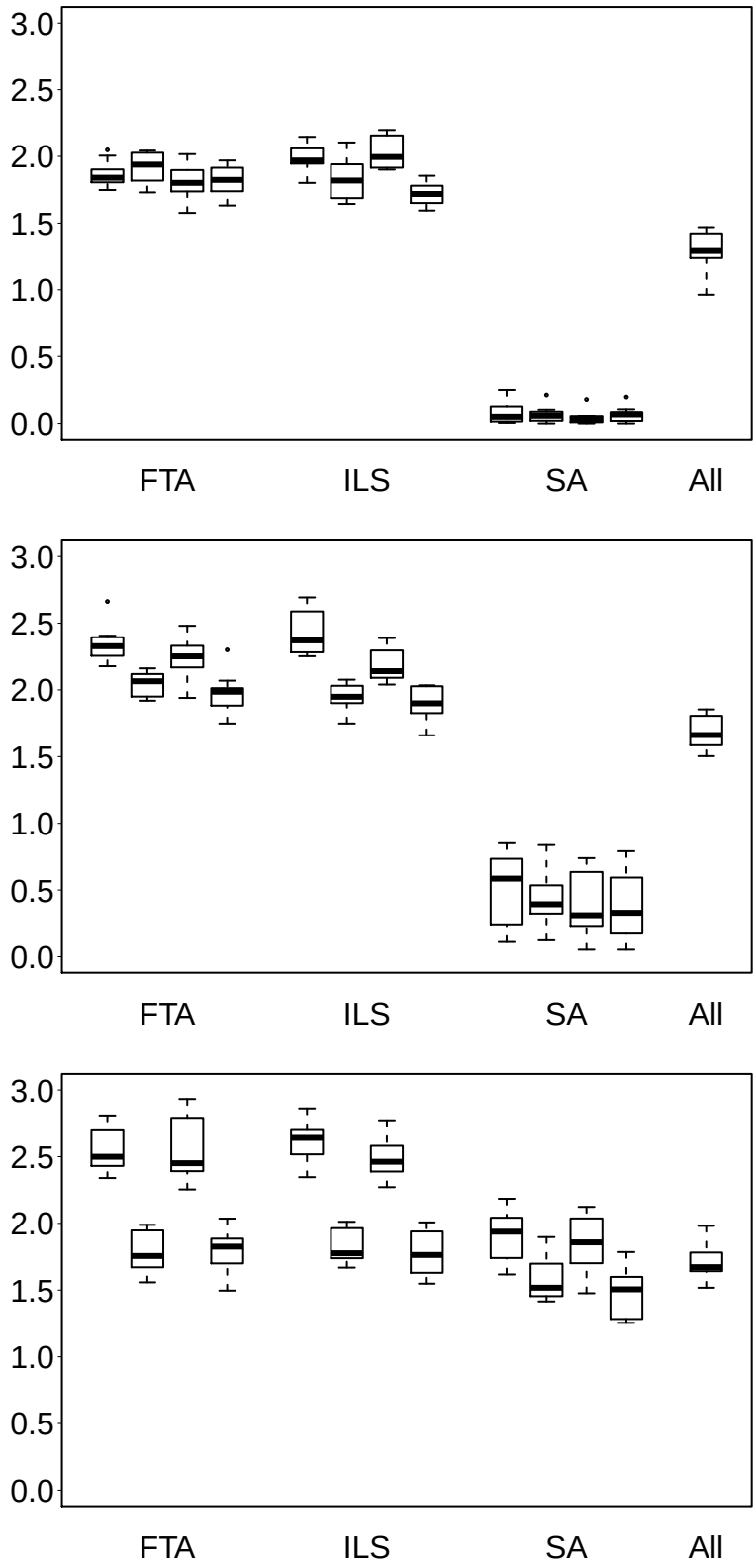

Figure 17: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Total Completion Time objective on the Taillard instances of sizes 50×5, 50×10, 50×20.

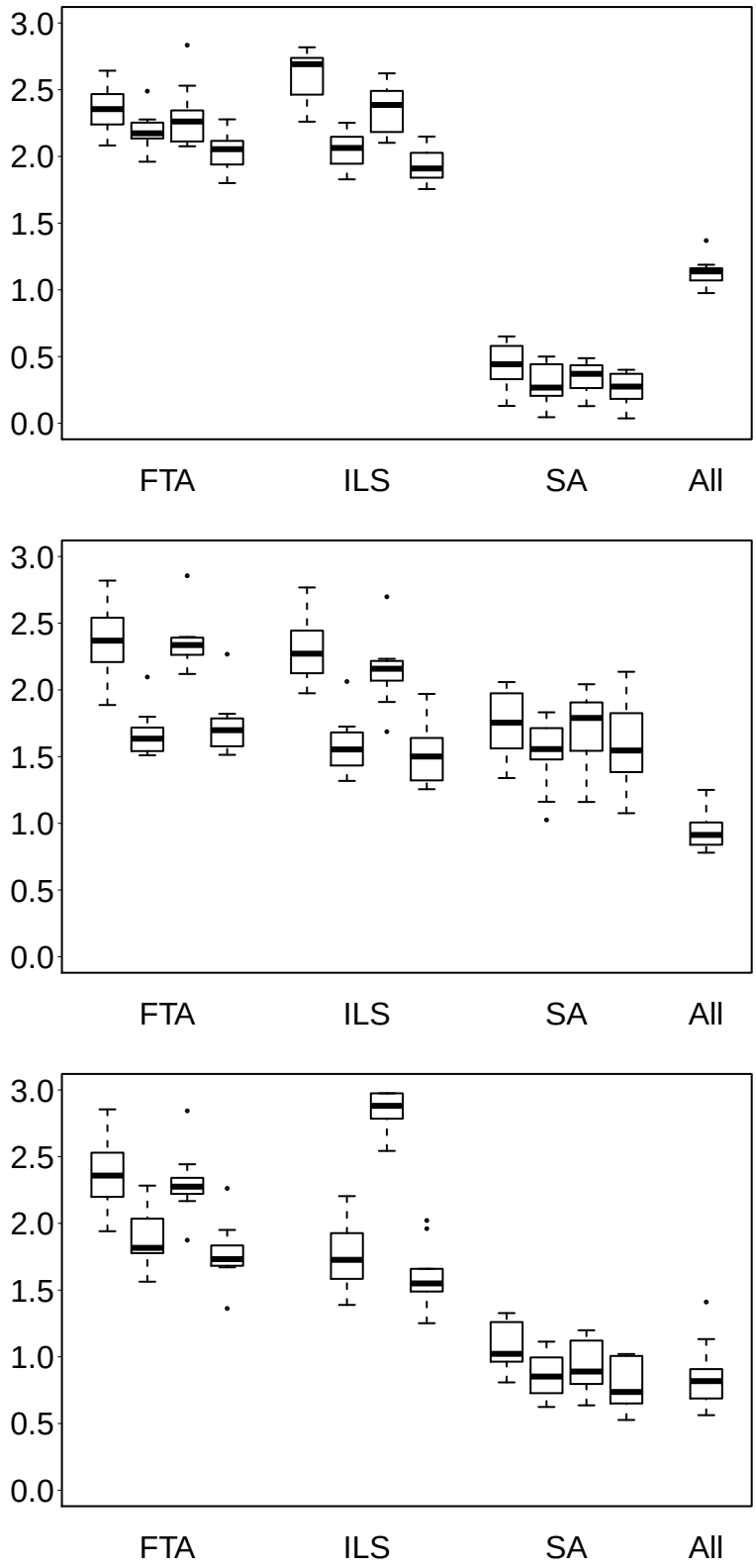

Figure 18: Comparison of the results obtained by FTA, a FTA alternated to a diversification mechanism, and SA, under the Total Completion Time objective on the Taillard instances of sizes 200×10, 200×20, 500×20.

References

[1] A. Franzin, T. Stützle, Revisiting simulated annealing: A component-based analysis, Computers & Operations Research 104 (2019) 191–206. doi:10.1016/j.cor.2018.12.015.