Supplementary Material to "Proactive and Reactive Constraint Programming for Stochastic Project Scheduling with Maximal Time-Lags"

Anonymous submission

Abstract

This document comprises the supplementary material (technical appendix) to the AAAI'25 submission "Proactive and Reactive Constraint Programming for Stochastic Project Scheduling with Maximal Time-Lags".

1 Tuning

This section describes the tuning process for the different scheduling methods. First, we investigated the effect of the time limit on solving the deterministic CP. Then, for the proactive approach, we tuned the time limit and the sample selection for the Sample Average Approximation (SAA). For the reactive approach, we tuned the quantile-approximation, and the time limit for rescheduling.

1.1 Time limit for CP on deterministic instances

To understand how the time limit may affect the results, we first consider the deterministic instances ofr RCPSP/max. Varakantham, Fu, and Lau (2016) reported on their time budget that *"$SORU$was able to obtain solutions within 5 minutes for every one instance in J10 and 2 hrs for the J20 instances. However, for J30, we were unable to get optimal solutions for certain instances in the cut-off limit of 3 hrs. On the other hand, $SORU-H$ was able to generate solutions for J10 instance within half of a second, J20 instances within 10 seconds and J30 instances within 10 minutes on average."*

In our research, we extend the instance sets with ubo50 and ubo100, for which the time limit can become more crucial. For each set (j10-ubo100), we fixed the first 50 instances from the PSPlib (Kolisch and Sprecher 1996) (PSP1 - PSP50). For j10-j30, the IBM CP Optimizer could solve all instances (PSP1-PSP50) within 60 seconds to optimality or prove infeasibility.

For the ubo50 instances PSP1 to PSP10 and the ubo100 instances PSP1 to PSP10, the effect of the time limits of 60 seconds, 600 seconds, and 3600 seconds respectively, is presented in Table 1 and 2. We observe that the solver status does not change when increasing the time limit to 600 seconds, and only flips 1 instance from feasible to infeasible when increasing the time limit from 600 seconds to 3600 seconds. The makespan only improves significantly for the ubo100 instances for higher time limits. In the remaining experiments, we fixed the time limit to 60 seconds for solving deterministic CPs.

Instance set	Time limit	Feasible	Infeasible	Optimal	Unkown
ubo50	60s	3	3	2	2
ubo50	600s	3	3	2	2
ubo50	3600s	2	3	3	2
ubo100	60s	4	5	0	1
ubo100	600s	4	5	0	1
ubo100	3600s	4	5	0	1

Table 1: Count per solver status IBM CP solver after different time limits on deterministic instances.

	60s	600s	3600s
ubo50	204	200	200
ubo100	421	410	410

Table 2: Average makespan (filtered feasible and optimal solutions on ubo50 and ubo100 set.

1.2 Proactive Approach

The sample selection process is expected to influence the performance of the proactive method. We used a subset of the j10 and ubo50 instances (for both sets PSP_1-PSP_20, and used 10 duration samples per instance. We fixed the time limit to 60 seconds and compared the feasibility ratios in Table 3. We found that the higher feasibility ratios were obtained using two settings for the proactive method, being 1 sample with $\gamma = 0.9$, to which we will refer with $\mathsf{proactive}_{0.9}$, and the setting with the smart samples to which we will refer with $\mathsf{proactive}_{\mathsf{SAA}}$ in the remaining of our experiments. For the SAA with four samples (smart samples), we investigated the effect of the time limit on the makespan in Table 4. We observed that the makespan would still improve while making the step from 600 seconds to 3600 seconds. We decide to use a time limit of 1800 seconds instead of 3600 seconds in the remaining of the experiments to be able to conduct more experiments.

1.3 Reactive Approach

First, we observed the effect of the duration estimations on the performance of the reactive approach. We used a sub-

1 sample	γ =0.5	γ =0.75	γ =0.9	γ =1
j10	0.11	0.53	0.94	0.75
ubo50	0	0.01	0.92	0.80
multiple samples	n=5	n=10	n=25	n=50
j10	0.63	0.78	0.94	0.94
ubo50	0.12	0.48	0.89	0.91
smart samples*	n=4			
j10	0.94			
ubo50	0.92			

Table 3: Feasibility ratio for different γ-quantiles in proactive approach. Each cell counts hits out of 200 experiments. * Uses a combination of smart samples that are the quantiles with $\gamma \in [0.25, 0.5, 0.75, 0.9]$

	60s	600s	3600s
ubo50	233	232	231
ubo100	485	480	472

Table 4: Average makespan obtained with proactive SAA with four scenarios for different time limits (filtered feasible and optimal solutions on ubo50 and ubo100 set.

set of the j10 and ubo50 instances (for both sets PSP_1-PSP_20, and used 10 duration samples per instance. We fixed the time limit for the initial schedule to 60 seconds and for the rescheduling to 2 seconds.

	γ =0.5	γ =0.75	γ =0.9	γ =1
j10	0.11	0.53	0.94	0.75
ubo50	0	0.01	0.92	0.80

Table 5: Feasibility ratio for different γ-quantiles in reactive approach. Each cell counts hits out of 200 experiments.

Remarkably, we observe similar feasibility ratios to the proactive approach, indicating that for feasibility the initial schedule is quite important. For the final evaluation, we fixed the reactive approach with $\gamma = 0.9$, and will analyze how the solution quality improves with the rescheduling procedure compared to a standard proactive approach.

Next, we observe the effect of the time limit for rescheduling. We used the same subset of the j10 and ubo50 instances (for both sets PSP_1-PSP_20, used 10 duration samples per instance, and runtime limits of 1, 2, 10 and 30 seconds. The results (in Table 6) are almost similar for the different time limits, this might be because the solver finishes already before the time limit, and the increase in time online has mainly to do with the number of solver calls. In the experimental evaluation, we therefore fixed the rescheduling time limit to 2 seconds, which we expected to be sufficient for larger or slightly more complicated problems.

	time limit rescheduling	1s	2s	10s	30s
j10	makespan	38	38	38	39
j10	time online	0.03	0.04	0.04	0.04
ubo50	makespan	171	171	171	172
ubo 50	time online	15.4	15.1	15.1	15.4

Table 6: Average makespan and time online for double hits on different time limits for rescheduling. Each cell averages over double hits out of 200 experiments.

1.4 Hyperparameters Selection

This subsection presents the hyperparameters in Table 7 that are used in the final experiments that are also presented in the main paper.

Hyperparameter	proactive$_{SAA}$	proactive$_{0.9}$	stnu	reactive
γ	[0.25, 0.5, 0.75, 0.7]	0.9	1	0.9
Time limit CP	1800s	600s	600s	600s and 2s
Solver	IBM CP	IBM CP	IBM CP	IBM CP

Table 7: Hyperparameter settings

2 Statistical Tests

2.1 Wilcoxon Test

The Wilcoxon test that we use follows Cureton (1967) and is described below:

- Collect a set of matched pairs (the results from two different methods on one instance sample).
- Compute the difference between the two test results for each pair.

  An important remark is that because of the discrete objective values (makespan), we can obtain a zero difference when there is a tie, we use Pratt's procedure for handling ties (Pratt 1959), which includes zero-differences in the ranking process, but drops the ranks of the zeros afterward.
- Order the pairs according to the absolute values of the differences.
- Assign ranks to the pairs based on these absolute values.
- Sum the positive ranks (T_{pos}) and the negative (T_{neg}) ranks separately.
- Take the smaller of the two $T = min(T_{pos}, T_{neg})$.
- If the two methods have no consistent difference in their relative performances, then the rank-sums should be approximately equal. This is tested with a normal approximation for the Wilcoxon statistic which is outlined by Cureton (1967). Cureton (1967) propose a corrected normal approximation which is needed because of usage of the the Pratt procedure for handling zero differences.
- The normalized Z-statistic is given by the formula: $z = (T - d)/se$, where d is the continuity correction from Cureton (1967), and se is the standard error.

All of the above can be executed using the Python package SciPy (Virtanen et al. 2020) built-in method $scipy.stats.wilcoxon$ that is called with parameters $method$="approx", "$zero\_method$="pratt", and $correction$=True.

2.2 Z-test for Proportion (Binomial Distribution)

We use (Kanji 2006) as a reference for the Z-test for Proportion. It is important to mention that this test is approximate as it assumes that the number of observations justifies a normal approximation for the binomial. (In contradiction to the SciPy package and its built-in method $scipy.stats.binomtest$ containing an exact test).

The proportion test investigates whether the is a significant difference between the assumed proportion of wins p_0 and the observed proportion of wins p. In our analysis, two methods are compared and the number of wins for each method is counted based on one metric.

The procedure for the proportion test is as follows:

- Collect a set of matched pairs (the results from two different methods on one atomic instance form a pair).
- For each pair, determine which method wins, and count the wins for both methods. Exclude all ties.
- Calculate the ratio of wins for one of the two methods.
- Test where this ratio differs significantly from $p_0 = 0.5$ (equal probability of winning).
  - The test statistic is $Z = \frac{|p-p_0|-\frac{1}{2n}}{\{\frac{p_0(1-p_0)}{n}^{1/2}\}}$
  - The term $\frac{1}{2n}$ in the numerator is a discontinuity correction.
  - For a two-sided test with a significance level $\alpha = 0.05$ the acceptance region for the null hypothesis is $-1.96 < Z < 1.96$.

.

2.3 Magnitude Test

The magnitude test we use is a t-test for two population means, or the method of paired comparisons such as Test 10 in the book by Kanji (2006). The test whether there is a significant difference between two population means. The procedure for this paired comparison t-test is as follows:

- Collect a set of matched pairs (so the results from two different methods on one atomic instance forms a pair).
- Normalise the performances for each pair by computing the mean value of the pair and dividing the two items in the pair by the pairs' mean such that all normalized observations will be between 0 and 2, and 1 indicates a tie.
- Compute the differences d_1 between the two test results for each pair i.
- Compute the variances of the differences with the following formula: $s^2 = \sum_{i=1}^{n} \frac{(d_i-\bar{d})^2}{n-1}$.
- Compute the means of both methods $\bar{x_1}$ and $\bar{x_2}$.
- Compute the t-statistic using the formula: $t = \frac{\bar{x_1}-\bar{x_2}}{s/n^{1/2}}$
- Test significance by checking whether t lies within the acceptance region for which the values are given by the Student's t-distribution (two-sided) with $n-1$ degrees of freedom.

After the normalization step, it is possible to execute the test with the Scipy package and specifically, its built-in method $scipy.stats.ttest\_rel$.

3 Results Tables

Please find the results of the statistical test in this section. This data led to the Figures that are included in our main paper.

- Tables 10 for noise level $c = 1$ and 12 for noise level $c = 2$: including the results of the Wilcoxon and proportion tests on solution quality.
- Tables 11 for noise level $c = 1$ and 13 for noise level $c = 2$: including the results of the magnitude test (independent t-test) on solution quality.
- Tables 14 for noise level $c = 1$ and 16 for noise level $c = 2$: including the results of the Wilcoxon and proportion tests on time offline.
- Tables 15 for noise level $c = 1$ and 17 for noise level $c = 2$: including the results of the magnitude test (independent t-test) on time offline.
- Tables 18 and Tables 20: including the results of the Wilcoxon and proportion tests on time online.
- Tables 19 and 21: including the results of the magnitude test (independent t-test) on time online.

3.1 Feasibility Ratio

First, we analyze the feasibility ratio obtained by the different methods. For $c = 1$, we observe the feasibility ratio obtained by $\text{proactive}_{\text{SAA}}$, $\text{proactive}_{0.9}$, reactive are the same for instance sets j10 - ubo50, and only for ubo100 the reactive approach is a little bit lower than the $\text{proactive}_{\text{SAA}}$, $\text{proactive}_{0.9}$. The stnu method has the lowest feasibility rate but remarkably the difference becomes smaller when the size of the problem grows (the difference is the smallest for ubo100).

We observe a different pattern for $c = 2$, with a higher noise factor. The highest feasibility ratios are obtained by stnu. This could be explained by the fact that the stnu method uses the information about the distribution and is, therefore, better at handling the larger variances in activity duration.

Set	stnu	$\text{proactive}_{\text{SAA}}$	$\text{proactive}_{0.9}$	reactive
j10	0.65	0.85	0.85	0.85
j20	0.65	0.76	0.76	0.76
j30	0.78	0.89	0.89	0.89
ubo50	0.77	0.86	0.86	0.86
ubo100	0.84	0.91	0.91	0.88

Table 8: Feasibility ratio for noise factor $c = 1$ We use abbreviations for the proactive approach with $\gamma = 0.9$ ($\text{pro}_{0.9}$), and the SAA variant (pro_{SAA}).

Set	stnu	$\text{proactive}_{\text{SAA}}$	$\text{proactive}_{0.9}$	reactive
j10	0.63	0.63	0.64	0.63
j20	0.63	0.54	0.53	0.53
j30	0.63	0.51	0.48	0.49
ubo50	0.67	0.41	0.42	0.41
ubo100	0.79	0.35	0.36	0.33

Table 9: Feasibility ratio for noise factor $c = 2$ We use abbreviations for the proactive approach with $\gamma = 0.9$ ($\text{pro}_{0.9}$), and the SAA variant (pro_{SAA}).

3.2 Partial Ordering Solution Quality

Next, we analyze the test results based on the Wilcoxon test, proportion test, and magnitude test for solution quality. We translate the test results into partial orderings that are visualized with solid arrows indicating a significant difference based on the Wilcoxon test and dashed arrows indicating a significant difference based on the proportion test for the pairs where the Wilcoxon test did not find significant results. Furthermore, for all pairs where we find a significant difference according to Wilcoxon, or the proportion test, we analyze whether we can find a significant magnitude difference on the double hits.

- First, we discuss partial orderings for the noise level $c = 1$, which are visualized in Figure 1. For $c = 1$, the partial ordering shows the same pattern across the different instance sets. However, there are a few exceptions:
  - In some cases we can only show a dashed arrow, indicating a weaker partial ordering, that is a significant proportional difference (proportion of wins) instead of a significant difference obtained by Wilcoxon.
  - Only for ubo50, all arrows are solid, indicating that for all pairs we found a significant difference according to the Wilcoxon test.
  - For ubo100, there is no significant difference between $\text{proactive}_{\text{SAA}}$ and $\text{proactive}_{0.9}$. Possibly for the larger instances the SAA becomes more difficult to solve, for which reason it is not better than the heuristic γ-quantile procedure anymore.
- Then, we analyze the partial orderings for the noise level $c = 2$, in Figure 2. Here there are a few things to observe:
  - We observe a shift in the partial orderings for $c = 2$.
  - For j10-j20, the partial ordering is similar to $c = 1$, although for $c = 2$ all arrows are solid (indicating a relation obtained by the Wilcoxon test).
  - For $j30$, there is no significant difference between $\text{proactive}_{0.9}$ and $\text{proactive}_{\text{SAA}}$, neither with Wilcoxon and neither proportionally.
  - The partial orderings for $j30$ and $ubo50$ are similar.
  - For ubo100, we observe the stnu is significantly better than the three other methods, although no significant difference can be found between the three other methods.
- The main conclusion is that in all situations the stnu shows to be the outperforming method based on solution quality. In general, the reactive shows to outperform the proactivemethods. Furthermore, $\text{proactive}_{\text{SAA}}$ outperforms in many cases $\text{proactive}_{0.9}$, although for larger instances and a higher noise level this difference is not present anymore.

3.3 Partial Ordering Time Offline

- We expected the $\text{proactive}_{0.9}$ and reactive to be the fastest offline, followed by the $\text{proactive}_{\text{SAA}}$ and stnu, where the partial ordering between the latter two depends on the problem size. When two methods have (almost) only ties (such as $\text{proactive}_{0.9}$ and reactive) not all tests can be executed and a nan will appear.
- Importantly, the tests are designed in such a way that infeasibilities are also weighted in significance testing. When a method results in an infeasible solution, infinite offline time is assigned to this experiment. For that reason, it can still occur that we find a difference between reactive and $\text{proactive}_{0.9}$: although they have the same offline procedure, one of the two can still fail which results in infinite offline time.
- In general, we give preference to the test results of the Wilcoxon test because this test shows a stronger relation than the proportional test, but to fill in the missing information we use the proportion test to test if there is a significant difference in the proportion of wins. We noted, however, that the test outcomes of the two tests can sometimes be contradicting. The Wilcoxon test penalizes infeasibilities more severely, while the proportion test only considers the number of wins. Consequently, it is pos-

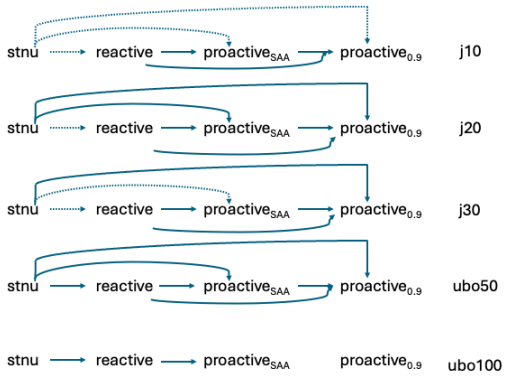

Figure 1: Partial ordering on solution quality for $c = 1$. Solid arrow indicating a significant difference obtained by the Wilcoxon test, and dashed arrow only a proportional difference.

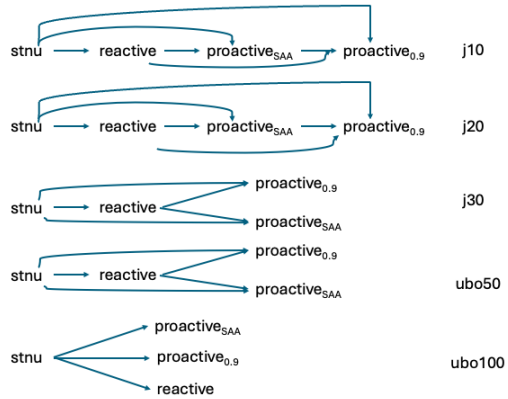

Figure 2: Partial ordering on solution quality for $c = 2$. Solid arrow indicating a significant difference obtained by the Wilcoxon test, and dashed arrow only a proportional difference.

sible that method 1 produces more infeasible solutions but achieves better metrics, whereas method 2 generates fewer infeasible solutions but worse metrics, resulting in the Wilcoxon test on instances where at least one feasible solution exists tells us method 2 performs better (infeasible solutions are penalized more heavily). Proportion test on instances where at least one feasible solution exists tells us method 1 performs better. The magnitude test on double hits tells us Method 1 performs better as it has better metric values.

- Now, we observe the different partial orderings obtained for $c = 1$:
  - In general, the time spent offline for the $\text{proactive}_{0.9}$ and the reactive is exactly the same, therefore in most cases we also did not find a significant difference between the two. However, especially for the ubo100 instances, we found that sometimes $\text{proactive}_{0.9}$ had more feasible solutions, resulting in a partial ordering of $\text{proactive}_{0.9} \rightarrow \text{reactive}$.
  - We observe that for j10 and j20 $\text{proactive}_{\text{SAA}}$ is consistently faster offline than stnu, but this flips for j30 - ubo100, where the stnu becomes faster. This can be explained by the fact that the solve time of the SAA can grow exponentially, while the DC-checking algorithm is a polynomial time algorithm.
- For noise level $c = 2$, we observe in Figure 4:
  - For j10 and j20, the partial ordering is almost the same as for the $c = 1$ level, although the flipping already occurs at j20, where the stnu becomes faster than the $\text{proactive}_{\text{SAA}}$.
  - Surprisingly, the pattern changes for the ubo50 and ubo100 instances. We observe that the best performance on time offline is obtained by the stnu according to the Wilcoxon. We found that this is mainly caused by the much higher feasibility ratio of the stnu method.
  - Furthermore, we find that the $\text{proactive}_{0.9}$ becomes better than reactive because of the higher feasibility ratio again (as the two methods have the same offline procedure).
  - Again, for the larger instances sets (j20-ubo100), the $\text{proactive}_{\text{SAA}}$ has the worst relative computation time offline.

3.4 Partial Ordering Time Online

We observe the results from the tests on online computation time:

- For $c = 1$ (Figure 6), we find the same partial ordering for each instance set, which is also in line with our expectations:
  - There is no difference between $\text{proactive}_{\text{SAA}}$ and $\text{proactive}_{0.9}$ as both only require a feasibility check only.
  - The stnu real-time execution algorithm turns out to be more efficient online than the reactive method, which comprises multiple solver calls, this is also expected.

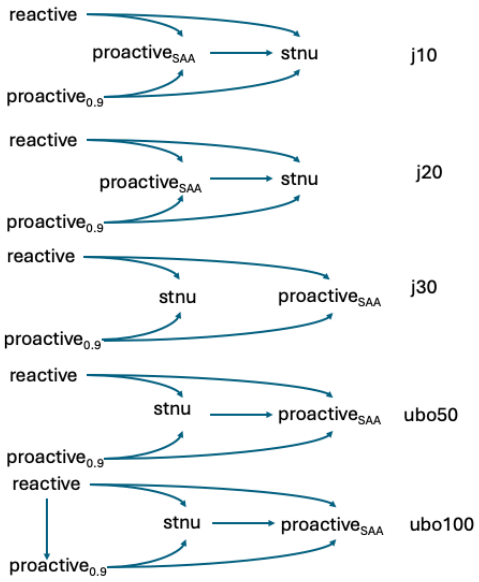

Figure 3: Partial ordering time offline c=1. Solid arrow indicating a significant difference obtained by the Wilcoxon test, and dashed arrow only a proportional difference.

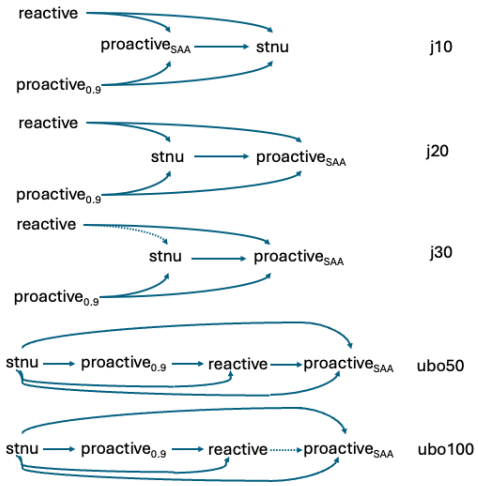

Figure 4: Partial ordering time offline c=2. Solid arrow indicating a significant difference obtained by the Wilcoxon test, and dashed arrow only a proportional difference.

- For $c = 2$ (Figure 6), we find the same pattern as for $c = 1$, for problem sets j10-j30, however, we find that for ubo50 and ubo100, the stnu starts to outperform other methods, explained by the higher feasibility ratio of the stnu.

3.5 Magnitude Tests

In general, we observed that for the pairs of methods for which we found a significant partial ordering, we also found a significant difference in the magnitude of the different metrics. The test results for solution quality are presented in Ta-

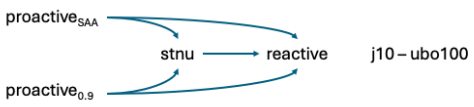

Figure 5: Partial ordering time online c=1, j10, j20, j30, ubo50, and ubo100. Solid arrow indicating a significant difference obtained by the Wilcoxon test, and dashed arrow only a proportional difference.

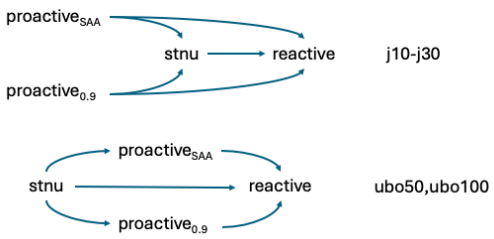

Figure 6: Partial ordering time online c=2, ubo50, and ubo100. Solid arrow indicating a significant difference obtained by the Wilcoxon test, and dashed arrow only a proportional difference.

ble 11 (c=1) and 13 (c=2), for time offline in Table 15 (c=1) and 17 (c=2).

The remarkable things that we observed in the results of the magnitude tests, were also already observed while comparing Wilcoxon test results with proportional test results in the earlier analysis for a few cases:

- For time offline, $c = 2$, ubo50 and ubo100, the Wilcoxon test found hat the stnu is better than e.g. proactive$_{0.9}$ and reactive, while looking at the magnitude results on double hits (this is an important difference in test procedure!), we find that the proactive$_{0.9}$ and reactive are much faster offline than stnu, which is also expected. Then, we observe no difference between proactive$_{0.9}$ and reactive, which can be explained by that fact that both have the same offline procedure and we select only double hits for the test.
- We find something similar occurs for the time online with $c = 2$, according to Wilcoxon stnu is outperforming proactive$_{SAA}$ and proactive$_{0.9}$ for ubo50 and ubo100. However, this is again mainly caused by the higher ratio of feasible solutions, as when we analyze the magnitude difference the proactive methods are much faster online looking at the double hits (which is logical as this comprises only of feasibility checking).

3.6 Summary

In our main paper, we employed the statistical analysis from this document to provide a summarizing partial ordering per metric including a brief summary of the main findings. We decide to present the most occuring patterns, and in text highlight any important exceptions.

4 Reproducibility

We uploaded our code to make it possible to reproduce the results of our research and to facilitate others using a similar benchmarking method. Please refer to the `READ.ME` of our repository to find instructions for requirements and directions to our experiment scripts, and the scripts to run the statistical tests.

References

Cureton, E. E. 1967. The normal approximation to the signed-rank sampling distribution when zero differences are present. *Journal of the American Statistical Association*, 62(319): 1068–1069.

Kanji, G. K. 2006. 100 statistical tests.

Kolisch, R.; and Sprecher, A. 1996. PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, 205–216.

Pratt, J. W. 1959. Remarks on zeros and ties in the Wilcoxon signed rank procedures. *Journal of the American Statistical Association*, 54(287): 655–667.

Varakantham, P.; Fu, N.; and Lau, H. 2016. A Proactive Sampling Approach to Project Scheduling under Uncertainty. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

j10	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[340] -1.659 (0.097)	[340] -0.067 (0.947)	[340] -0.082 (0.935)	[340] -10.571 (*)	[340] -10.918 (*)	[340] -5.756 (*)
	[308] 0.591 (*)	[310] 0.632 (*)	[310] 0.632 (*)	[117] 0.991 (*)	[121] 1.0 (*)	[37] 0.973 (*)
j20	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[270] -1.523 (0.128)	[270] -2.401 (*)	[270] -2.679 (*)	[270] -5.936 (*)	[270] -7.852 (*)	[270] -6.144 (*)
	[213] 0.596 (*)	[215] 0.633 (*)	[213] 0.643 (*)	[75] 0.827 (*)	[62] 1.0 (*)	[59] 0.898 (*)
j30	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[320] -0.892 (0.372)	[320] -1.218 (0.223)	[320] -2.157 (*)	[320] -5.54 (*)	[320] -9.395 (*)	[320] -5.299 (*)
	[214] 0.579 (*)	[214] 0.575 (*)	[217] 0.618 (*)	[103] 0.757 (*)	[89] 1.0 (*)	[79] 0.797 (*)
ubo50	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[370] -6.748 (*)	[370] -6.8 (*)	[370] -6.859 (*)	[370] -2.76 (*)	[370] -8.531 (*)	[370] -7.085 (*)
	[277] 0.776 (*)	[276] 0.779 (*)	[278] 0.781 (*)	[61] 0.672 (*)	[73] 1.0 (*)	[65] 0.938 (*)
ubo100	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[400] -2.12 (*)	[400] -1.76 (0.078)	[400] -1.842 (0.065)	[400] -0.442 (0.659)	[400] -6.534 (*)	[400] -0.2 (0.842)
	[312] 0.519 (0.533)	[314] 0.51 (0.778)	[311] 0.514 (0.65)	[147] 0.51 (0.869)	[88] 0.864 (*)	[151] 0.503 (1.0)

Table 10: Pairwise comparison on schedule quality (makespan) for noise factor c=1. Using a Wilcoxon test and a proportion test. Including all instances for which at least one of the two methods found a feasible solution. Note that the ordering matters: the first method showed is the better of the two in each paier method 1 - method 2. Each cell shows on the first row [nr pairs] z-value (p-value) of the Wilcoxon test with (*) for $p < 0.05$. Each cell shows on the second row [nr pairs] z-value (p-value) with (*) for $p < 0.05$

j10	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[260] -9.367 (*)	[260] -13.762 (*)	[260] -13.703 (*)	[340] -10.114 (*)	[340] -10.542 (*)	[340] -5.637 (*)
	stnu: 0.974	stnu: 0.962	stnu: 0.962	react: 0.986	react: 0.985	pro$_{SAA}$: 0.999
	react: 1.026	pro$_{SAA}$: 1.038	pro$_{0.9}$: 1.038	pro$_{SAA}$: 1.014	pro$_{0.9}$: 1.015	pro$_{0.9}$: 1.001
j20	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[230] -7.62 (*)	[230] -8.608 (*)	[230] -8.992 (*)	[270] -6.868 (*)	[270] -7.713 (*)	[270] -5.82 (*)
	stnu: 0.982	stnu: 0.978	stnu: 0.977	react: 0.995	react: 0.993	pro$_{SAA}$: 0.998
	react: 1.018	pro$_{SAA}$: 1.022	pro$_{0.9}$: 1.023	pro$_{SAA}$: 1.005	pro$_{0.9}$: 1.007	pro$_{0.9}$: 1.002
j30	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[280] -4.072 (*)	[280] -5.946 (*)	[280] -7.06 (*)	[320] -5.698 (*)	[320] -9.422 (*)	[320] -6.226 (*)
	stnu: 0.993	stnu: 0.99	stnu: 0.988	react: 0.997	react: 0.995	pro$_{SAA}$: 0.998
	react: 1.007	pro$_{SAA}$: 1.01	pro$_{0.9}$: 1.012	pro$_{SAA}$: 1.003	pro$_{0.9}$: 1.005	pro$_{0.9}$: 1.002
ubo50	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[330] -11.359 (*)	[330] -11.391 (*)	[330] -11.346 (*)	[370] -2.858 (*)	[370] -6.734 (*)	[370] -6.014 (*)
	stnu: 0.985	stnu: 0.985	stnu: 0.984	react: 1.0	react: 0.999	pro$_{SAA}$: 0.999
	react: 1.015	pro$_{SAA}$: 1.015	pro$_{0.9}$: 1.016	pro$_{SAA}$: 1.0	pro$_{0.9}$: 1.001	pro$_{0.9}$: 1.001
ubo100	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[358] -6.316 (*)	[370] -6.813 (*)	[370] -6.817 (*)	[388] -2.833 (*)	[388] -7.738 (*)	[400] -0.514 (0.608)
	stnu: 0.991	stnu: 0.99	stnu: 0.99	react: 0.999	react: 0.999	pro$_{SAA}$: 1.0
	react: 1.009	pro$_{SAA}$: 1.01	pro$_{0.9}$: 1.01	pro$_{SAA}$: 1.001	pro$_{0.9}$: 1.001	pro$_{0.9}$: 1.0

Table 11: Magnitude test on solution quality for noise factor c=1. Using a pairwise t-test, including all instances for which both methods found a feasible solution. Each cell shows on the first row [nr pairs] t-stat (p-value) with (*) for $p < 0.05$ and on the second row the normalized average of method 1 and on the third row the normalized average of method 2.

j10	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[258] -5.194 (*)	[258] -7.825 (*)	[258] -7.822 (*)	[235] -10.082 (*)	[234] -11.046 (*)	[235] -4.058 (*)
	[237] 0.696 (*)	[247] 0.802 (*)	[248] 0.802 (*)	[146] 0.911 (*)	[134] 0.993 (*)	[37] 0.838 (*)
j20	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[217] -9.825 (*)	[217] -10.194 (*)	[216] -10.57 (*)	[182] -4.58 (*)	[177] -7.243 (*)	[182] -3.241 (*)
	[185] 0.854 (*)	[188] 0.872 (*)	[188] 0.888 (*)	[79] 0.759 (*)	[53] 1.0 (*)	[60] 0.7 (*)
j30	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[233] -6.986 (*)	[234] -7.175 (*)	[232] -7.993 (*)	[183] -1.355 (0.175)	[163] -7.557 (*)	[182] -5.689 (*)
	[191] 0.749 (*)	[192] 0.776 (*)	[196] 0.796 (*)	[94] 0.574 (0.18)	[58] 1.0 (*)	[96] 0.802 (*)
ubo50	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[284] -11.76 (*)	[285] -11.73 (*)	[284] -11.799 (*)	[171] -1.973 (*)	[162] -3.74 (*)	[173] -0.938 (0.348)
	[236] 0.907 (*)	[238] 0.908 (*)	[236] 0.911 (*)	[50] 0.66 (*)	[39] 0.821 (*)	[36] 0.583 (0.405)
ubo100	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[285] -12.139 (*)	[284] -11.55 (*)	[288] -11.26 (*)	[141] -1.83 (0.067)	[122] -1.017 (0.309)	[146] -0.083 (0.934)
	[261] 0.851 (*)	[254] 0.799 (*)	[265] 0.796 (*)	[82] 0.415 (0.151)	[44] 0.477 (0.88)	[82] 0.5 (0.912)

Table 12: Pairwise comparison on schedule quality (makespan) for noise factor c=2. Using a Wilcoxon test and a proportion test. Including all instances for which at least one of the two methods found a feasible solution. Note that the ordering matters: the first method showed is the better of the two in each pair method 1 - method 2. Each cell shows on the first row [nr pairs] z-value (p-value) of the Wilcoxon test with (*) for $p < 0.05$. Each cell shows on the second row [nr pairs] z-value (p-value) with (*) for $p < 0.05$ for the proportion test.

j10	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[205] -9.128 (*)	[204] -16.872 (*)	[206] -17.395 (*)	[230] -10.858 (*)	[233] -11.811 (*)	[231] -4.856 (*)
	stnu: 0.963	stnu: 0.929	stnu: 0.925	react: 0.966	react: 0.964	pro$_{SAA}$: 0.997
	react: 1.037	pro$_{SAA}$: 1.071	pro$_{0.9}$: 1.075	pro$_{SAA}$: 1.034	pro$_{0.9}$: 1.036	pro$_{0.9}$: 1.003
j20	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[170] -10.855 (*)	[172] -11.237 (*)	[170] -11.89 (*)	[174] -4.95 (*)	[176] -6.22 (*)	[173] -3.656 (*)
	stnu: 0.972	stnu: 0.968	stnu: 0.964	react: 0.991	react: 0.989	pro$_{SAA}$: 0.997
	react: 1.028	pro$_{SAA}$: 1.032	pro$_{0.9}$: 1.036	pro$_{SAA}$: 1.009	pro$_{0.9}$: 1.011	pro$_{0.9}$: 1.003
j30	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[140] -4.151 (*)	[148] -6.017 (*)	[138] -6.149 (*)	[152] -4.199 (*)	[160] -6.746 (*)	[150] -5.404 (*)
	stnu: 0.99	stnu: 0.986	stnu: 0.985	react: 0.994	react: 0.991	pro$_{SAA}$: 0.996
	react: 1.01	pro$_{SAA}$: 1.014	pro$_{0.9}$: 1.015	pro$_{SAA}$: 1.006	pro$_{0.9}$: 1.009	pro$_{0.9}$: 1.004
ubo50	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[141] -7.974 (*)	[146] -8.139 (*)	[148] -8.125 (*)	[145] -3.581 (*)	[155] -4.646 (*)	[150] -2.823 (*)
	stnu: 0.984	stnu: 0.981	stnu: 0.979	react: 0.998	react: 0.997	pro$_{SAA}$: 0.999
	react: 1.016	pro$_{SAA}$: 1.019	pro$_{0.9}$: 1.021	pro$_{SAA}$: 1.002	pro$_{0.9}$: 1.003	pro$_{0.9}$: 1.001
ubo100	stnu-react	stnu-pro$_{SAA}$	stnu-pro$_{0.9}$	react-pro$_{SAA}$	react-pro$_{0.9}$	pro$_{SAA}$-pro$_{0.9}$
	[94] -0.305 (0.761)	[114] -0.296 (0.767)	[114] -0.93 (0.355)	[76] -1.573 (0.12)	[99] -4.187 (*)	[94] 0.59 (0.556)
	stnu: 0.999	stnu: 0.999	stnu: 0.998	react: 0.999	react: 0.999	pro$_{SAA}$: 1.0
	react: 1.001	pro$_{SAA}$: 1.001	pro$_{0.9}$: 1.002	pro$_{SAA}$: 1.001	pro$_{0.9}$: 1.001	pro$_{0.9}$: 1.0

Table 13: Magnitude test on solution quality for noise factor c=2. Using a pairwise t-test, including all instances for which both methods found a feasible solution. Each cell shows on the first row [nr pairs] t-stat (p-value) with (*) for $p < 0.05$ and on the second row the normalized average of method 1 and on the third row the normalized average of method 2.

j10	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	pro$_{SAA}$-stnu
	[340] nan (nan)	[340] -15.982 (*)	[340] -15.982 (*)	[340] -15.982 (*)	[340] -15.982 (*)	[340] -13.986 (*)
	[nan] nan (nan)	[340] 1.0 (*)	[340] 1.0 (*)	[340] 1.0 (*)	[340] 1.0 (*)	[340] 0.882 (*)
j20	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	pro$_{SAA}$-stnu
	[270] nan (nan)	[270] -14.125 (*)	[270] -14.245 (*)	[270] -14.125 (*)	[270] -14.245 (*)	[270] -3.927 (*)
	[nan] nan (nan)	[270] 0.963 (*)	[270] 1.0 (*)	[270] 0.963 (*)	[270] 1.0 (*)	[270] 0.63 (*)
j30	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	stnu-pro$_{SAA}$
	[320] nan (nan)	[320] -13.963 (*)	[320] -15.506 (*)	[320] -13.963 (*)	[320] -15.506 (*)	[320] -1.63 (0.103)
	[nan] nan (nan)	[320] 0.969 (*)	[320] 1.0 (*)	[320] 0.969 (*)	[320] 1.0 (*)	[320] 0.5 (0.955)
ubo50	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	stnu-pro$_{SAA}$
	[370] nan (nan)	[370] -16.671 (*)	[370] -16.671 (*)	[370] -16.671 (*)	[370] -16.671 (*)	[370] -7.261 (*)
	[nan] nan (nan)	[370] 1.0 (*)	[370] 1.0 (*)	[370] 1.0 (*)	[370] 1.0 (*)	[370] 0.622 (*)
ubo100	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	stnu-pro$_{SAA}$
	[400] -3.463 (*)	[400] -14.988 (*)	[400] -15.286 (*)	[400] -15.969 (*)	[400] -17.332 (*)	[400] -6.406 (*)
	[12] 1.0 (*)	[400] 0.962 (*)	[400] 0.97 (*)	[400] 0.975 (*)	[400] 1.0 (*)	[400] 0.6 (*)

Table 14: Pairwise comparison on time offline for noise factor c=1. Using a Wilcoxon test and a proportion test. Including all instances for which at least one of the two methods found a feasible solution. Note that the ordering matters: the first method showed is the better of the two in each pair method 1 - method 2 according to Wilcoxon. Each cell shows on the first row [nr pairs] z-value (p-value) of the Wilcoxon test with (*) for $p < 0.05$. Each cell shows on the second row [nr pairs] ratio of wins (p-value) with (*) for $p < 0.05$

j10	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	pro$_{SAA}$-stnu
	[340] nan (nan)	[260] -365.854 (*)	[340] -77.22 (*)	[260] -365.854 (*)	[340] -77.22 (*)	[260] -19.399 (*)
	pro$_{0.9}$: 1.0	react: 0.14	react: 0.32	pro$_{0.9}$: 0.14	pro$_{0.9}$: 0.32	pro$_{SAA}$: 0.65
	react: 1.0	stnu: 1.86	pro$_{SAA}$: 1.68	stnu: 1.86	pro$_{SAA}$: 1.68	stnu: 1.35
j20	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	pro$_{SAA}$-stnu
	[270] nan (nan)	[230] -45.769 (*)	[270] -52.707 (*)	[230] -45.769 (*)	[270] -52.707 (*)	[230] -3.062 (*)
	pro$_{0.9}$: 1.0	react: 0.22	react: 0.28	pro$_{0.9}$: 0.22	pro$_{0.9}$: 0.28	pro$_{SAA}$: 0.91
	react: 1.0	stnu: 1.78	pro$_{SAA}$: 1.72	stnu: 1.78	pro$_{SAA}$: 1.72	stnu: 1.09
j30	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	stnu-pro$_{SAA}$
	[320] nan (nan)	[280] -38.703 (*)	[320] -53.836 (*)	[280] -38.703 (*)	[320] -53.836 (*)	[280] 0.5 (0.618)
	pro$_{0.9}$: 1.0	react: 0.3	react: 0.32	pro$_{0.9}$: 0.3	pro$_{0.9}$: 0.32	stnu: 1.02
	react: 1.0	stnu: 1.7	pro$_{SAA}$: 1.68	stnu: 1.7	pro$_{SAA}$: 1.68	pro$_{SAA}$: 0.98
ubo50	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	stnu-pro$_{SAA}$
	[370] nan (nan)	[330] -36.15 (*)	[370] -72.563 (*)	[330] -36.15 (*)	[370] -72.563 (*)	[330] -5.636 (*)
	pro$_{0.9}$: 1.0	react: 0.36	react: 0.24	pro$_{0.9}$: 0.36	pro$_{0.9}$: 0.24	stnu: 0.82
	react: 1.0	stnu: 1.64	pro$_{SAA}$: 1.76	stnu: 1.64	pro$_{SAA}$: 1.76	pro$_{SAA}$: 1.18
ubo100	pro$_{0.9}$-react	react-stnu	react-pro$_{SAA}$	pro$_{0.9}$-stnu	pro$_{0.9}$-pro$_{SAA}$	stnu-pro$_{SAA}$
	[388] nan (nan)	[358] -27.209 (*)	[388] -70.349 (*)	[370] -25.293 (*)	[400] -70.926 (*)	[370] -6.391 (*)
	pro$_{0.9}$: 1.0	react: 0.43	react: 0.27	pro$_{0.9}$: 0.45	pro$_{0.9}$: 0.27	stnu: 0.79
	react: 1.0	stnu: 1.57	pro$_{SAA}$: 1.73	stnu: 1.55	pro$_{SAA}$: 1.73	pro$_{SAA}$: 1.21

Table 15: Magnitude test on time offline for noise factor c=1. Using a pairwise t-test, including all instances for which both methods found a feasible solution. Each cell shows on the first row [nr pairs] t-stat (p-value) with (*) for $p < 0.05$ and on the second row the normalized average of method 1 and on the third row the normalized average of method 2.

j10	$pro_{0.9}$-react	react-stnu	react-pro_{SAA}	$pro_{0.9}$-stnu	$pro_{0.9}$-pro_{SAA}	pro_{SAA}-stnu
	[234] -0.996 (0.319)	[258] -9.382 (*)	[235] -12.848 (*)	[258] -9.554 (*)	[235] -13.069 (*)	[258] -6.664 (*)
	[nan] nan (nan)	[258] 0.903 (*)	[235] 0.991 (*)	[258] 0.907 (*)	[235] 0.996 (*)	[258] 0.764 (*)
j20	$pro_{0.9}$-react	react-stnu	react-pro_{SAA}	$pro_{0.9}$-stnu	$pro_{0.9}$-pro_{SAA}	stnu-pro_{SAA}
	[177] -0.994 (0.32)	[217] -4.543 (*)	[182] -10.458 (*)	[216] -4.456 (*)	[182] -10.213 (*)	[217] -5.281 (*)
	[nan] nan (nan)	[217] 0.816 (*)	[182] 0.973 (*)	[216] 0.815 (*)	[182] 0.967 (*)	[217] 0.604 (*)
j30	$pro_{0.9}$-react	react-stnu	react-pro_{SAA}	$pro_{0.9}$-stnu	$pro_{0.9}$-pro_{SAA}	stnu-pro_{SAA}
	[163] -1.728 (0.084)	[233] -0.083 (0.934)	[183] -7.137 (*)	[232] -0.267 (0.789)	[182] -6.667 (*)	[234] -7.974 (*)
	[3] 0.0 (0.248)	[233] 0.7 (*)	[183] 0.891 (*)	[232] 0.69 (*)	[182] 0.879 (*)	[234] 0.726 (*)
ubo50	stnu-$pro_{0.9}$	stnu-react	stnu-pro_{SAA}	$pro_{0.9}$-react	$pro_{0.9}$-pro_{SAA}	react-pro_{SAA}
	[284] -4.827 (*)	[284] -5.557 (*)	[285] -10.588 (*)	[162] -2.643 (*)	[173] -8.741 (*)	[171] -7.484 (*)
	[284] 0.454 (0.138)	[284] 0.479 (0.514)	[285] 0.712 (*)	[7] 1.0 (*)	[173] 0.936 (*)	[171] 0.906 (*)
ubo100	stnu-$pro_{0.9}$	stnu-react	stnu-pro_{SAA}	$pro_{0.9}$-react	$pro_{0.9}$-pro_{SAA}	react-pro_{SAA}
	[288] -8.783 (*)	[285] -10.58 (*)	[284] -13.068 (*)	[122] -4.786 (*)	[146] -5.008 (*)	[141] -1.441 (0.15)
	[288] 0.576 (*)	[285] 0.653 (*)	[284] 0.82 (*)	[23] 1.0 (*)	[146] 0.836 (*)	[141] 0.702 (*)

Table 16: Pairwise comparison on time offline for noise factor c=2. Using a Wilcoxon test and a proportion test. Including all instances for which at least one of the two methods found a feasible solution. Note that the ordering matters: the first method showed is the better of the two in each pair method 1 - method 2 according to Wilcoxon. Each cell shows on the first row [nr pairs] z-value (p-value) of the Wilcoxon test with (*) for $p < 0.05$. Each cell shows on the second row [nr pairs] ratio of wins (p-value) with (*) for $p < 0.05$

j10	$pro_{0.9}$-react	react-stnu	react-pro_{SAA}	$pro_{0.9}$-stnu	$pro_{0.9}$-pro_{SAA}	pro_{SAA}-stnu
	[233] nan (nan)	[205] -229.689 (*)	[230] -59.39 (*)	[206] -230.812 (*)	[231] -59.404 (*)	[204] -17.066 (*)
	$pro_{0.9}$: 1.0	react: 0.15	react: 0.35	$pro_{0.9}$: 0.15	$pro_{0.9}$: 0.35	pro_{SAA}: 0.61
	react: 1.0	stnu: 1.85	pro_{SAA}: 1.65	stnu: 1.85	pro_{SAA}: 1.65	stnu: 1.39
j20	$pro_{0.9}$-react	react-stnu	react-pro_{SAA}	$pro_{0.9}$-stnu	$pro_{0.9}$-pro_{SAA}	stnu-pro_{SAA}
	[176] nan (nan)	[170] -82.607 (*)	[174] -63.732 (*)	[170] -82.607 (*)	[173] -63.382 (*)	[172] 0.379 (0.705)
	$pro_{0.9}$: 1.0	react: 0.21	react: 0.23	$pro_{0.9}$: 0.21	$pro_{0.9}$: 0.23	stnu: 1.01
	react: 1.0	stnu: 1.79	pro_{SAA}: 1.77	stnu: 1.79	pro_{SAA}: 1.77	pro_{SAA}: 0.99
j30	$pro_{0.9}$-react	react-stnu	react-pro_{SAA}	$pro_{0.9}$-stnu	$pro_{0.9}$-pro_{SAA}	stnu-pro_{SAA}
	[160] nan (nan)	[140] -33.765 (*)	[152] -40.336 (*)	[138] -33.253 (*)	[150] -39.904 (*)	[148] -2.168 (*)
	$pro_{0.9}$: 1.0	react: 0.34	react: 0.33	$pro_{0.9}$: 0.34	$pro_{0.9}$: 0.33	stnu: 0.9
	react: 1.0	stnu: 1.66	pro_{SAA}: 1.67	stnu: 1.66	pro_{SAA}: 1.67	pro_{SAA}: 1.1
ubo50	stnu-$pro_{0.9}$	stnu-react	stnu-pro_{SAA}	$pro_{0.9}$-react	$pro_{0.9}$-pro_{SAA}	react-pro_{SAA}
	[148] 26.653 (*)	[141] 27.168 (*)	[146] -0.952 (0.343)	[155] nan (nan)	[150] -45.457 (*)	[145] -44.29 (*)
	stnu: 1.7	stnu: 1.72	stnu: 0.95	$pro_{0.9}$: 1.0	$pro_{0.9}$: 0.24	react: 0.24
	$pro_{0.9}$: 0.3	react: 0.28	pro_{SAA}: 1.05	react: 1.0	pro_{SAA}: 1.76	pro_{SAA}: 1.76
ubo100	stnu-$pro_{0.9}$	stnu-react	stnu-pro_{SAA}	$pro_{0.9}$-react	$pro_{0.9}$-pro_{SAA}	react-pro_{SAA}
	[114] 14.765 (*)	[94] 15.183 (*)	[114] -3.125 (*)	[99] nan (nan)	[94] -42.151 (*)	[76] -42.944 (*)
	stnu: 1.55	stnu: 1.61	stnu: 0.83	$pro_{0.9}$: 1.0	$pro_{0.9}$: 0.21	react: 0.18
	$pro_{0.9}$: 0.45	react: 0.39	pro_{SAA}: 1.17	react: 1.0	pro_{SAA}: 1.79	pro_{SAA}: 1.82

Table 17: Magnitude test on time offline for noise factor c=2. Using a pairwise t-test, including all instances for which both methods found a feasible solution. Each cell shows on the first row [nr pairs] t-stat (p-value) with (*) for $p < 0.05$ and on the second row the normalized average of method 1 and on the third row the normalized average of method 2.

j10	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[340] -0.368 (0.713)	[340] -15.993 (*)	[340] -15.997 (*)	[340] -15.98 (*)	[340] -15.98 (*)	[340] -2.726 (*)
	[85] 0.471 (0.664)	[340] 1.0 (*)	[340] 1.0 (*)	[340] 1.0 (*)	[340] 1.0 (*)	[340] 0.765 (*)
j20	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[270] -1.17 (0.242)	[270] -14.246 (*)	[270] -14.245 (*)	[270] -14.243 (*)	[270] -14.243 (*)	[270] -6.441 (*)
	[133] 0.556 (0.225)	[270] 1.0 (*)	[270] 1.0 (*)	[270] 1.0 (*)	[270] 1.0 (*)	[270] 0.852 (*)
j30	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[320] -0.653 (0.514)	[320] -15.505 (*)	[320] -15.505 (*)	[320] -15.504 (*)	[320] -15.504 (*)	[320] -8.247 (*)
	[191] 0.524 (0.563)	[320] 1.0 (*)	[320] 1.0 (*)	[320] 1.0 (*)	[320] 1.0 (*)	[320] 0.875 (*)
ubo50	pro_{SAA}-$pro_{0.9}$	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[370] -0.771 (0.44)	[370] -16.67 (*)	[370] -16.67 (*)	[370] -16.67 (*)	[370] -16.67 (*)	[370] -9.859 (*)
	[299] 0.522 (0.488)	[370] 1.0 (*)	[370] 1.0 (*)	[370] 1.0 (*)	[370] 1.0 (*)	[370] 0.892 (*)
ubo100	pro_{SAA}-$pro_{0.9}$	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[400] -0.719 (0.472)	[400] -17.331 (*)	[400] -17.331 (*)	[400] -17.331 (*)	[400] -17.331 (*)	[400] -12.488 (*)
	[391] 0.453 (0.069)	[400] 1.0 (*)	[400] 1.0 (*)	[400] 1.0 (*)	[400] 1.0 (*)	[400] 0.925 (*)

Table 18: Pairwise comparison on time online for noise factor c=1. Using a Wilcoxon test and a proportion test. Including all instances for which at least one of the two methods found a feasible solution. Note that the ordering matters: the first method showed is the better of the two in each pair method 1 - method 2. Each cell shows on the first row [nr pairs] z-value (p-value) of the Wilcoxon test with (*) for $p < 0.05$. Each cell shows on the second row [nr pairs] proportion (p-value) with (*) for $p < 0.05$

j10	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[340] 0.679 (0.498)	[260] -109.014 (*)	[260] -106.917 (*)	[340] -4393.475 (*)	[340] -4277.995 (*)	[260] -1151.565 (*)
	$pro_{0.9}$: 1.02	$pro_{0.9}$: 0.05	pro_{SAA}: 0.05	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.04
	pro_{SAA}: 0.98	stnu: 1.95	stnu: 1.95	react: 2.0	react: 2.0	react: 1.96
j20	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[270] -0.663 (0.508)	[230] -285.563 (*)	[230] -282.464 (*)	[270] -7311.952 (*)	[270] -7470.984 (*)	[230] -605.38 (*)
	$pro_{0.9}$: 0.97	$pro_{0.9}$: 0.03	pro_{SAA}: 0.03	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.07
	pro_{SAA}: 1.03	stnu: 1.97	stnu: 1.97	react: 2.0	react: 2.0	react: 1.93
j30	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[320] -0.156 (0.876)	[280] -789.246 (*)	[280] -786.726 (*)	[320] -14638.229 (*)	[320] -14425.009 (*)	[280] -338.664 (*)
	$pro_{0.9}$: 0.99	$pro_{0.9}$: 0.02	pro_{SAA}: 0.02	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.09
	pro_{SAA}: 1.01	stnu: 1.98	stnu: 1.98	react: 2.0	react: 2.0	react: 1.91
ubo50	pro_{SAA}-$pro_{0.9}$	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[370] -0.795 (0.427)	[330] -3951.584 (*)	[330] -3976.441 (*)	[370] -30723.506 (*)	[370] -31447.766 (*)	[330] -142.758 (*)
	pro_{SAA}: 0.97	$pro_{0.9}$: 0.01	pro_{SAA}: 0.01	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.15
	$pro_{0.9}$: 1.03	stnu: 1.99	stnu: 1.99	react: 2.0	react: 2.0	react: 1.85
ubo100	pro_{SAA}-$pro_{0.9}$	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[400] -0.065 (0.948)	[370] -56517.571 (*)	[370] -48810.689 (*)	[388] -83435.86 (*)	[388] -78013.468 (*)	[358] -60.202 (*)
	pro_{SAA}: 1.0	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.23
	$pro_{0.9}$: 1.0	stnu: 2.0	stnu: 2.0	react: 2.0	react: 2.0	react: 1.77

Table 19: Magnitude t-test on time online for noise factor c=1 (double hits). Each cell shows on the first row [nr pairs] t-stat (p-value) with (*) for $p < 0.05$ and on the second row the normalized average of method 1 and on the third row the normalized average of method 2.

j10	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[235] -0.384 (0.701)	[258] -9.554 (*)	[258] -9.212 (*)	[234] -13.262 (*)	[235] -12.623 (*)	[258] -8.801 (*)
	[66] 0.47 (0.712)	[258] 0.907 (*)	[258] 0.899 (*)	[234] 1.0 (*)	[235] 0.987 (*)	[258] 0.891 (*)
j20	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[182] -1.204 (0.229)	[216] -4.456 (*)	[217] -4.913 (*)	[177] -11.279 (*)	[182] -10.957 (*)	[217] -11.457 (*)
	[79] 0.43 (0.261)	[216] 0.815 (*)	[217] 0.825 (*)	[177] 0.994 (*)	[182] 0.984 (*)	[217] 0.968 (*)
j30	pro_{SAA}-$pro_{0.9}$	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[182] -2.014 (*)	[232] -0.928 (0.353)	[234] -2.526 (*)	[163] -10.268 (*)	[183] -9.31 (*)	[233] -9.839 (*)
	[119] 0.58 (0.099)	[232] 0.69 (*)	[234] 0.735 (*)	[163] 0.982 (*)	[183] 0.94 (*)	[233] 0.901 (*)
ubo50	$pro_{0.9}$-pro_{SAA}	stnu-$pro_{0.9}$	stnu-pro_{SAA}	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[173] -0.722 (0.47)	[284] -3.843 (*)	[285] -3.933 (*)	[162] -11.04 (*)	[171] -9.02 (*)	[284] -13.107 (*)
	[147] 0.524 (0.621)	[284] 0.43 (*)	[285] 0.435 (*)	[162] 1.0 (*)	[171] 0.942 (*)	[284] 0.951 (*)
ubo100	pro_{SAA}-$pro_{0.9}$	stnu-$pro_{0.9}$	stnu-pro_{SAA}	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[146] -0.482 (0.63)	[288] -8.467 (*)	[284] -9.06 (*)	[122] -9.585 (*)	[141] -6.136 (*)	[285] -14.285 (*)
	[144] 0.458 (0.359)	[288] 0.576 (*)	[284] 0.585 (*)	[122] 1.0 (*)	[141] 0.837 (*)	[285] 0.982 (*)

Table 20: Pairwise comparison on time online for noise factor c=2. Using a Wilcoxon test and a proportion test. Including all instances for which at least one of the two methods found a feasible solution. Note that the ordering matters: the first method showed is the better of the two in each paier method 1 - method 2. Each cell shows on the first row [nr pairs] z-value (p-value) of the Wilcoxon test with (*) for $p < 0.05$. Each cell shows on the second row [nr pairs] proportion (p-value) with (*) for $p < 0.05$

j10	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[231] 0.391 (0.697)	[206] -92.546 (*)	[204] -91.532 (*)	[233] -4426.055 (*)	[230] -4653.225 (*)	[205] -1399.699 (*)
	$pro_{0.9}$: 1.01	$pro_{0.9}$: 0.06	pro_{SAA}: 0.06	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.03
	pro_{SAA}: 0.99	stnu: 1.94	stnu: 1.94	react: 2.0	react: 2.0	react: 1.97
j20	$pro_{0.9}$-pro_{SAA}	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[173] 0.77 (0.443)	[170] -241.897 (*)	[172] -262.612 (*)	[176] -7787.819 (*)	[174] -8817.279 (*)	[170] -935.848 (*)
	$pro_{0.9}$: 1.03	$pro_{0.9}$: 0.03	pro_{SAA}: 0.03	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.06
	pro_{SAA}: 0.97	stnu: 1.97	stnu: 1.97	react: 2.0	react: 2.0	react: 1.94
j30	pro_{SAA}-$pro_{0.9}$	$pro_{0.9}$-stnu	pro_{SAA}-stnu	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[150] -0.264 (0.792)	[138] -514.302 (*)	[148] -547.413 (*)	[160] -14437.976 (*)	[152] -14871.238 (*)	[140] -397.488 (*)
	pro_{SAA}: 0.99	$pro_{0.9}$: 0.02	pro_{SAA}: 0.02	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.07
	$pro_{0.9}$: 1.01	stnu: 1.98	stnu: 1.98	react: 2.0	react: 2.0	react: 1.93
ubo50	$pro_{0.9}$-pro_{SAA}	stnu-$pro_{0.9}$	stnu-pro_{SAA}	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[150] -0.707 (0.481)	[148] 2600.902 (*)	[146] 2585.661 (*)	[155] -25795.773 (*)	[145] -23210.69 (*)	[141] -121.392 (*)
	$pro_{0.9}$: 0.96	stnu: 1.99	stnu: 1.99	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.14
	pro_{SAA}: 1.04	$pro_{0.9}$: 0.01	pro_{SAA}: 0.01	react: 2.0	react: 2.0	react: 1.86
ubo100	pro_{SAA}-$pro_{0.9}$	stnu-$pro_{0.9}$	stnu-pro_{SAA}	$pro_{0.9}$-react	pro_{SAA}-react	stnu-react
	[94] 0.141 (0.888)	[114] 21743.706 (*)	[114] 21911.639 (*)	[99] -46429.42 (*)	[76] -40238.643 (*)	[94] -41.405 (*)
	pro_{SAA}: 1.0	stnu: 2.0	stnu: 2.0	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	stnu: 0.21
	$pro_{0.9}$: 1.0	$pro_{0.9}$: 0.0	pro_{SAA}: 0.0	react: 2.0	react: 2.0	react: 1.79

Table 21: Magnitude t-test on time online for noise factor c=2 (double hits). Each cell shows on the first row [nr pairs] t-stat (p-value) with (*) for $p < 0.05$ and on the second row the normalized average of method 1 and on the third row the normalized average of method 2.