Evaluating Proactive, Reactive, and Hybrid Strategies for the Stochastic Multi-Mode RCPSP with Hard Deadlines

Andreas Shiamishis

Supervisors: Kim van den Houten, Léon Planken, Mathijs de Weerdt

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering

June 22, 2025

Final project course: CSE3000 Research Project

1 Abstract

Stochastic scheduling is a crucial and rapidly growing field that attracts significant interest across numerous domains, particularly in the development of digital factories. We evaluate and compare three algorithms for the stochastic Multi-Mode Resource Constrained Project Scheduling Problem with Hard Deadlines. We outline the proactive, reactive and hybrid approaches and compare their performance in terms of feasibility, execution time and makespan. We also experiment with higher variance, different instance sizes and duration distributions to understand the factors that affect performance. Our results highlight the potential of the hybrid approach and demonstrate the fundamental trade-off between robustness and schedule quality.

2 Introduction

Scheduling is one of the most pertinent challenges of the industrial era. From distributing assignments in a team, to creating a plan of execution for product development and production, scheduling concerns all efficiency-driven companies and agents. Throughout the years, many deterministic and randomized algorithms have been developed to solve these problems while experimenting with the trade-off between optimality and runtime.

However, the uncertainty of the real world adds another level of complexity to this NP-hard problem [1]. With probabilistic and uncertain durations of tasks, more thorough research is required to test and evaluate algorithms. There has been a considerable amount of research for deterministic problems [4], but not as extensive for the stochastic version. The aim of this project is to explore the utility of Simple Temporal Networks with Uncertainty (or STNUs for short) [11], in solving instances of Resource Constraint Project Scheduling Problems (or RCPSP for short) with deadlines and comparing it to other state-of-the-art algorithms.

A recent study [6] proposed the use of STNUs with the RTE* algorithm [7] to provide good approximate solutions to the stochastic RCPSP. This algorithm utilizes a Partial Order Schedule (or POS for short) that is created offline, and adapts it accordingly during execution. This hybrid approach was then compared with simple proactive and reactive methods and concluded that the hybrid, STNU method outperformed the basic proactive and reactive algorithms in terms of solution quality (measured by makespan).

We extend this study [6] by analyzing the STNU-based approach for instances of the Multi-Mode Resource Constrained Project Scheduling Problem (or MMRCPSP for short) where some tasks have hard deadlines. Our findings indicate that the hybrid STNU-based approach can provide high-quality schedules with high feasibility ratios. Combining it with robust sampling consistently yields Pareto-optimal results, as does the proactive method with 0.5 quantile sampling. Among these, proactive-0.5 achieves better makespan, illustrating the tradeoff between feasibility and optimality.

3 Related Work

There has been a lot of research in the fields of algorithmic and stochastic scheduling. Ramos et al. [12] formulated the Multi-Mode Resource Constraint Project Scheduling Problem and proposed a multi-start iterated local search approach to solve it. Hartmann and Drexl [5] provide a comprehensive comparison of exact branch-and-bound algorithms for the MMRCPSP, introducing a new enumeration strategy and evaluating it against existing methods using theoretical analysis and extensive computational experiments. Bold et al. [2] proposed

a mixed-integer linear programming approach for solving the stochastic MMR-CPSP. Van den Houten et al. [6] proposed the use of simple temporal networks with uncertainty to approach the stochasticity of the problem and provided an in depth comparison between this new method and two baseline ones, a proactive and a reactive algorithm. This research addresses the gap in existing work by incorporating multiple modes and hard deadlines and exploring the effectiveness of STNUs in solving this scheduling problem.

4 Methodology and Background

4.1 Problem Definition

The Multi-Mode Resource Constrained Project Scheduling Problem with Hard Deadlines can be expanded from [12] as follows:

- A set of tasks $T = \{0, 1, \ldots, n+1\}$, where 0 is the dummy source and $n+1$ the dummy sink task.
- For each task $i \in T \setminus \{0, n+1\}$, a set of execution modes $M_i = \{1, \ldots, m_i\}$.
- A set of renewable resources $R = \{1, \ldots, r\}$ with capacities c_k for each $k \in R$.
- Each mode $m \in M_i$ for activity i specifies:
  - A stochastic duration $D_{i,m}$.
  - Resource consumption $r^k_{i,m}$ for each resource k.
- A set of precedence constraints $(i, j) \in P \subseteq T \times T$, indicating the strict order of execution between tasks. These include four types:
  - $P_{\text{SBS}} \subseteq P$: StartBeforeStart — task i must start before task j starts
  - $P_{\text{SBE}} \subseteq P$: StartBeforeEnd — task i must start before task j ends
  - $P_{\text{EBS}} \subseteq P$: EndBeforeStart — task i must end before task j starts
  - $P_{\text{EBE}} \subseteq P$: EndBeforeEnd — task i must end before task j ends
- Hard deadlines d_i for tasks $T_d \in T$.

Decision Variables

- $x_{i,m} \in \{0, 1\}$ 1 if activity i is executed in mode m, 0 otherwise.
- $S_i \in \mathbb{R}_{\geq 0}$ Start time of activity i
- $D_i = \sum_{m \in M_i} x_{i,m} D_{i,m}$ Stochastic duration of activity i
- $E_i = S_i + D_i$ End time of activity i

Constraints

- $\sum_{m \in M_i} x_{i,m} = 1 \, \forall i \in T \setminus \{0, n+1\}$ One mode per task

- StartBeforeStart (SBS): $S_i \leq S_j \quad \forall (i, j) \in P_{\text{SBS}}$ Start of i before start of j
- StartBeforeEnd (SBE): $S_i \leq E_j \quad \forall (i, j) \in P_{\text{SBE}}$ Start of i before end of j
- EndBeforeStart (EBS): $E_i \leq S_j \quad \forall (i, j) \in P_{\text{EBS}}$ End of i before start of j
- EndBeforeEnd (EBE): $E_i \leq E_j \quad \forall (i, j) \in P_{\text{EBE}}$ End of i before end of j
- $\sum_{i \in J} \sum_{m \in M_i} x_{i,m} \cdot r^k_{i,m} \cdot \mathbb{I}_{[S_i, E_i)}(t) \leq c_k \forall k \in R, \forall t$ Resource limits

- $E_i \leq d_i \forall i \in T_d$ Deadline adherence

Objectives

- $\min E_{n+1}$ Minimize makespan

4.2 Stochastic Scheduling Algorithms

Scheduling under durational uncertainty has led to the development of various algorithmic approaches, which can be broadly positioned along a spectrum from fully proactive to fully reactive strategies.

Before introducing the algorithms, it is crucial to present the sampling strategies considered, as they are relevant for all algorithms. We have experimented with the following:

- **Robust**: Select the worst-case scenario, assuming each activity takes its maximum possible duration.
- **Mean-based**: Use the expected value of each duration distribution.
- **Quantile-based**: Choose a specific quantile (e.g., 0.9 quantile) from each duration distribution.

The choice of sampling method influences the trade-off between schedule quality and schedule feasibility. More pessimistic samples (e.g., robust or high quantile samples) typically yield higher feasibility ratios but poorer performance in terms of makespan. In contrast, optimistic samples (e.g., using means or low quantile samples) can produce schedules with shorter expected makespans, but they are more likely to become infeasible during execution [6].

4.2.1 The Proactive Approach

The proactive algorithm we evaluated in this research aims to construct a schedule in advance and strictly follow it during execution, without adjusting to real-time events. This relies on transforming the stochastic problem into a deterministic one by sampling from the duration distributions of tasks as explained above.

4.2.2 The Reactive Approach

The reactive algorithm we evaluated in this paper, dynamically adjusts the schedule in response to real-time information (e.g. completion of tasks) during execution. It begins with an initial deterministic schedule based on a chosen estimate of the activity durations, just like in the proactive approach (e.g., mean, robust or quantile).

During execution, whenever an activity finishes, the algorithm checks whether the actual completion time differs from the estimated time. If so, it triggers a full rescheduling procedure by solving a new MMRCPSP instance.

4.2.3 The STNU based approach

The aforementioned algorithms will be evaluated against an STNU-based approach that utilizes both proactive scheduling and dynamic adjustments [6]. This approach creates a Partial Order Schedule (or POS for short) which consists of precedence relations and temporal constraints but maintains temporal flexibility in terms of actual starting times of tasks. The POS is modelled using an STNU, which is then checked for dynamic controllability [10] to ensure that the schedule can adjust in real time to uncertain events regardless of how they unfold. During the actual execution, as tasks are completed and their real durations are revealed, the starting times of the unscheduled tasks are determined based on the POS.

4.3 Methodology

We outline key aspects of our process, including key assumptions and decisions.

4.3.1 Modelling the Problem

To model deadlines in the Partial Order Schedule, we added dummy tasks representing each deadline as separate jobs with zero resource consumption, start time at zero and fixed duration equal to the deadline. These were linked via precedence constraints to ensure all associated tasks completed before the dummy deadline task ended. This indirect modelling enforces deadline adherence during execution from the POS, without interfering with resource constraints. One such a representation is found in appendix C.

4.3.2 Mode selection

We perform mode selection offline for all algorithms by solving the deterministic CP model with sampled durations. This simplifies execution and reduces online runtime, as mode flexibility is not required

during online rescheduling and thus the number of variables and constraints decreases.

4.3.3 Evaluation metrics

To evaluate and compare the proactive, hybrid and reactive strategies, we use four metrics:

- **Feasibility ratio:** The proportion of simulated scenarios in which the schedule remains feasible, that is, all tasks are completed within their deadlines and without violating resource or temporal constraints.
- **Offline execution time:** The time required to compute the initial schedule before execution begins. This reflects the planning effort and computational complexity of the scheduling method.
- **Online execution time:** The time required to make real-time decisions during execution, such as adapting the schedule in response to observed delays or uncertainty realizations.
- **Makespan:** The total time required to complete all tasks in the project. A lower makespan indicates higher scheduling efficiency.

4.3.4 Distributions

In order to get a thorough understanding about the nature of the problem and the performance of the three approaches it is crucial to consider different distributions for the uncertain durations. This will cover a larger basis of real-world applications. For this research, we use the uniform and binomial distributions that relate to the noise factor as follows:

1. **Uniform Distribution:** The noise factor α determines the lower and upper bounds of the uniform distribution using the formula:lower bound $= \max(1,\ d - \alpha \cdot \sqrt{d})$, upper bound $= d + \alpha \cdot \sqrt{d}$, where d is the duration of the task.
2. **Binomial Distribution:** The same bounds are computed as in the uniform case. The number of trials is set as $n = \text{upper} - \text{lower}$, and durations are sampled from a binomial distribution with parameters n and fixed probability $p = 0.5$, then shifted by the lower bound.

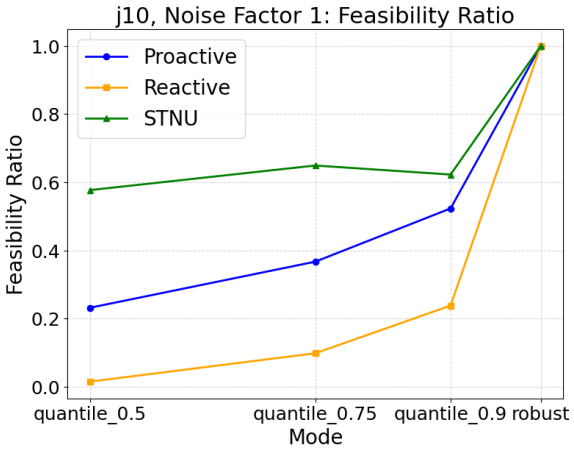
5 Experimental Process

The goal of our experiments is to understand the capabilities of these algorithms and how they compare with each other under varying conditions and levels of uncertainty.

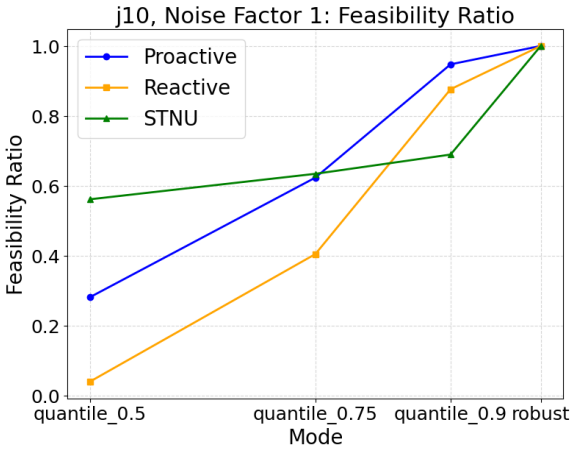
For this evaluation, we use instances from PSPLIB [8], which provides a well-established benchmark for project scheduling. We also add deadlines for some tasks in each sample in order to create instances for our specific problem. The range of deadlines is $d \in [m, h]$, where d is the deadline, m is the minimum completion time (taken as the minimum duration of any mode for a task) and h is the project horizon, the maximum completion time for all tasks (it is not considered in this project). The deadline generation uses the filename as the random seed. This results in some infeasible instances and some trivial ones, for example when deadlines are far away, but overall produces interpretable results since we use the same instances for all methods.

We use PyJobShop [9] for both modelling and solving deterministic reductions of instances. The algorithms are evaluated in instances with varying number of tasks and varying noise factors, which affect the uncertainty of scheduling. This gives insight not only on algorithms' performance but also on the areas they excel at.

Finally, we run the experiments on the DelftBlue supercomputer at TU Delft. The exact parameter settings that we used in the experiments can be found in Appendix A.



(a) Uniform distribution



(b) Binomial distribution

Figure 1: Feasibility ratios across three approaches. Results are based on 100 instances with 10 tasks and a noise factor of 1.

6 Results and Discussion

We now outline the most important findings and provide an analysis of the results.

6.1 Evaluation Metrics

In this section we analyze the results on each metric individually. We omit analysis of the offline times since, as expected, the STNU requires longer setup time before execution as, along with solving a deterministic constraint-programming model, it creates a partial order schedule. For completeness we have included the corresponding graph in appendix B. Offline execution time is also, arguably, less important than online execution time as most agents will have limited time to adjust their schedule online but will likely be more willing to spend more time offline to create a schedule.

6.1.1 Feasibility Ratio

Feasibility ratio is a key metric that reflects the likelihood of an algorithm producing a feasible schedule, that is, one that satisfies all imposed constraints. Evaluating and comparing feasibility ratios across different sampling methods is crucial, as a fully robust approach may guarantee 100% feasibility but often at the cost of suboptimal schedules with respect to makespan.

Figures 1a and 1b show the feasibility ratios for all methods under the uniform and binomial sampling distributions, respectively. Under the uniform distribution, the hybrid STNU-based approach consistently outperforms the two baselines across all sampling methods.

This performance can be attributed to the temporal flexibility offered by the hybrid approach, which allows it to dynamically adapt schedules and handle extreme task durations more effectively, especially when durations approach their maximum possible values.

In the case of the binomial distribution, the STNU-based method still performs best for lower quantile sampling (e.g., 75th and 50th percentiles), though it shows a larger dip in the higher quantile region (e.g., 90th percentile). Notably, the feasibility ratios of the STNU-based method remain relatively stable across both distributions, reinforcing the hypothesis that it is a robust and generalizable method.

The main difference between the two distributions lies in how the baseline methods behave. With the binomial distribution, where values above a certain

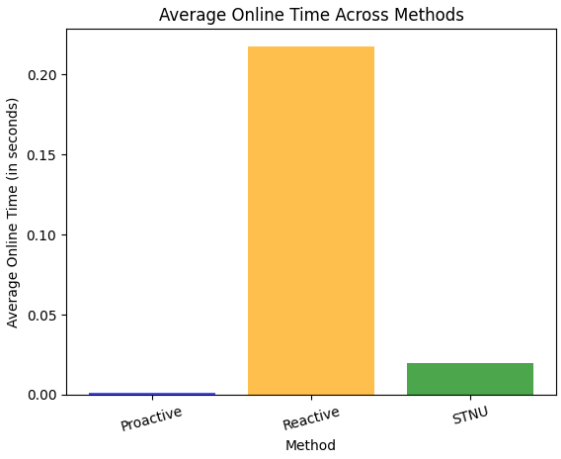


Figure 2: Average online execution times across the three approaches. Results are based on 100 instances, each evaluated with 10 scenarios, under two noise factors (1 and 2) and two instance sizes (10 and 20).

quantile tend to be closer to that quantile, the proactive and reactive algorithms suffer a less dramatic drop in feasibility as the sampling quantile decreases. This shows that the distribution choice strongly affects algorithm performance and can change how methods rank in terms of feasibility. Additional plots for other settings are included in Appendix B.

6.1.2 Online Execution Time

Online execution time is a critical consideration when evaluating scheduling algorithms, especially in real-world scenarios where decisions must be made quickly and adjustment time is limited. Excessive overhead during execution can become a bottleneck, undermining the practicality of even high-quality schedules.

The online behavior of the three approaches differs significantly, as explained in section 4.2. As shown in figure 2, the reactive approach incurs the highest online execution time, while the proactive method is the most efficient in this regard. The STNU-based approach, although slightly more demanding than the proactive algorithm due to its real-time decision-making, remains significantly more efficient than the reactive alternative.

Methods\Quantile	0.5	0.75	0.9	1
Reactive-Proactive	Proactive	Reactive	Reactive	Reactive
Reactive-STNU	STNU	STNU	STNU	STNU
Proactive-STNU	Proactive	STNU	STNU	STNU

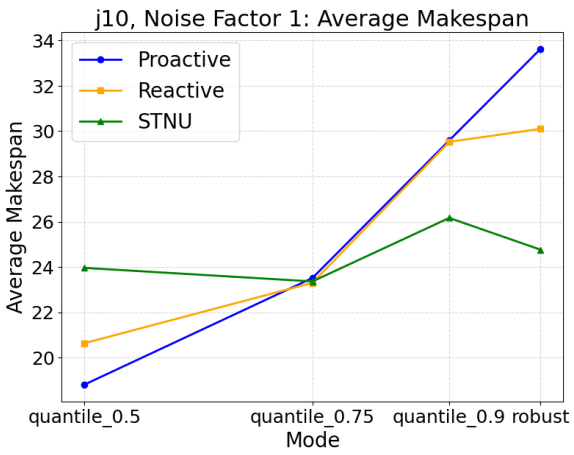
Table 1: Results of the Wilcoxon Matched-Pairs Rank-Sum Test between methods across different sampling quantiles for the binomial distribution. Each value considers all noise factor-instance size combinations. Cells in red indicate comparisons with non-significant differences (p-value > 0.05), using the Wilcoxon signed-rank test with $\alpha = 0.05$.

6.1.3 Makespan

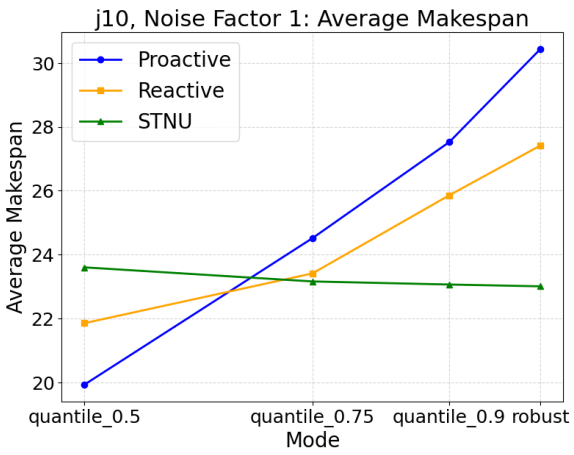
Finally, it is critical to present and analyze one of the most important and universal metrics in scheduling, the makespan. While each domain demands specific metrics and optimization objectives, minimizing the makespan of a schedule is often an important and central goal.

The figures 3a and 3b show the makespans for the uniform and binomial distributions respectively. It can be observed, that the STNU-based approach outperforms both the proactive and reactive approach in the robust mode by a relatively substantial margin. However, as the sampling method becomes more optimistic about the actual durations, the makespan of the other two approaches drops significantly while the value for the hybrid one experiences much less variance. This can be explained by the nature of the algorithms, since the hybrid approach does not rely on the sampling method as much since it only uses the sample for the partial order schedule.

When analyzing the results using this metric alone, the proactive approach with a mean-based sampling method seems to be the optimal algorithm. Despite the promising results, it is vital to under-



(a) Uniform distribution



(b) Binomial distribution

Figure 3: Makespan values across three approaches. Results are based on 100 instances with 10 tasks and a noise factor of 1.

stand that these values were derived only from feasible schedules, that is, schedules that satisfy the constraints. Since lower-quantile sampling methods will only produce feasible schedules for favorable, lower real durations, analysis should be performed for only the common instances. One test that can provide more insight in this regard is the Wilcoxon Matched-Pairs Rank-Sum Test [3], which compares the absolute differences between the results of the two methods and ranks them. This is useful in our case since it will give a clear indication to which approach outperforms the others, based only on common instances.

The results of the Wilcoxon test are presented in Table 1. These findings align with Figure 3, further illustrating the advantage of the proactive method over both the reactive and hybrid approaches when mean-based sampling is applied. Interestingly, the hybrid algorithm shows better performance than the reactive one, although the difference may not reach strong statistical significance. For a detailed view of the exact p-values, which provide further insight into the statistical relevance of the observed differences, we refer the reader to Table 2 in the appendix.

6.2 Cross-Metric Evaluation

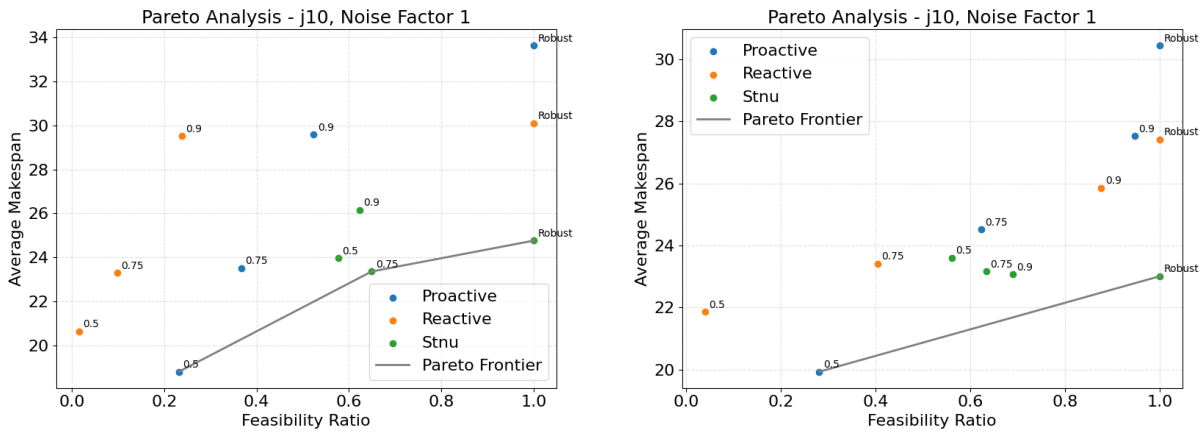
We will now look at the results from a more general point of view rather than analysing each metric individually.

6.2.1 Sampling

From the results above it is quite clear that the sampling used has a dramatic effect in the feasibility ratio and makespan for the proactive and reactive approaches, while it only seems to affect the feasibility ratio for the hybrid one. Consequently, robust sampling is optimal for the hybrid method as it ensures higher feasibility without compromising model performance.

6.2.2 Pareto Analysis

To combine the feasibility ratio and makespan and draw reasonable conclusions about the performance of each algorithm, we now present and analyze the pareto frontier as shown in 4 for both distributions. Looking at the graphs it becomes clear that there is no silver bullet that produces better schedules at a higher rate. On the other hand, we can see that, while some sampling method-algorithm combinations outperform oth-



(a) Uniform distribution

(b) Binomial distribution

Figure 4: Scatter plots of feasibility ratio against makespan for all sampling methods and both uniform and binomial distributions respectively. The grey line indicates the pareto optimal frontier.

ers, there is a tradeoff between the two metrics.

More specifically, in both distributions, the STNU-robust and proactive-0.5 approaches appear in the frontier, one offering optimal feasibility while the other generating better schedules. The remaining plots can be found in appendix B.

It is important to note here that the makespan results could be a bit misleading because the proactive approach is likely only generating feasible schedules for better samples from the distributions, while the hybrid one gives a feasible schedule independent of how the durations turn out. To really understand which of the two approaches creates higher quality schedules we run the Wilcoxon test on the two when using the same real durations.

For both distributions we find that the proactive-0.5 is indeed outperforming the STNU-robust algorithm based on makespan. The exact averages can be found in the appendix B for reference.

7 Conclusion and Future Work

In this paper we compare different approaches for the stochastic Multi-Mode Resource Constrained Project Scheduling Problem with Hard Deadlines. More specifically, we evaluate reactive, proactive and hybrid (STNU-based) algorithms on four metrics, feasibility, online and offline execution times and schedule quality, measured by makespan. We find that the hybrid approach with robust sampling gives the best tradeoff between feasibility and makespan without introducing large online computational overhead. The final choice of method depends on the specific characteristics of the problem and the scheduler's expectations and requirements.

In future work, other variations of these algorithms could be considered. For example, mode selection could occur during execution, increasing the flexibility of reactive and hybrid approaches. Furthermore, different modelling strategies for deadlines could be implemented and examined, as in our research we only focused on dummy task-based representations.

Responsible Research

The codebase used to generate these results along with the values of each individual run are available at `https://github.`

com/kimvandenhouten/PyJobShopSTNUs. To reproduce our experiments, one can run `python pyjobshop_pipeline.py mmrcpspd` and adjust the parameters in the `pyjobshop_pipeline.py` file accordingly. While this research aimed to approximate real-world scenarios as much as possible, it is important to acknowledge the assumptions made in the process. Fixed task dependencies, perfect knowledge of stochastic distributions, and immediate rescheduling capabilities without penalty were assumed. Additionally, all resource and mode-related uncertainties are assumed to be identical across tasks. These assumptions simplify analysis but may limit applicability to complex, dynamic settings.

While these algorithms require relatively low computational times for small instances, the execution times and computational demands increase exponentially with the complexity of the problems. This amount of resource consumption can have a significant environmental impact and should be considered when making decisions concerning these algorithms.

Furthermore, these methods are designed for efficiency and optimality and completely neglect the human aspect of job scheduling. They should be used with caution when there is a direct effect on employees as they might lead to overworking and exhaustion.

References

[1] M. Bartusch, R. H. Möhring, and F. J. Radermacher. "Scheduling project networks with resource constraints and time windows". In: *Annals of Operations Research* 16.1 (1988), pp. 199–240. ISSN: 1572-9338. DOI: 10.1007/BF02283745. URL: https://doi.org/10.1007/BF02283745.

[2] Matthew Bold and Marc Goerigk. *A faster exact method for solving the robust multi-mode resource-constrained project scheduling problem*. 2022. arXiv: 2203.06983 [math.OC]. URL: https://arxiv.org/abs/2203.06983.

[3] Edward E. Cureton. "The Normal Approximation to the Signed-Rank Sampling Distribution When Zero Differences are Present". In: *Journal of the American Statistical Association* 62.319 (1967), pp. 1068–1069. DOI: 10.1080/01621459.1967.10500917. eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1967.10500917. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1967.10500917.

[4] Sönke Hartmann and Dirk Briskorn. "A survey of variants and extensions of the resource-constrained project scheduling problem". In: *European Journal of Operational Research* 207.1 (2010), pp. 1–14. ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2009.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0377221709008558.

[5] Sönke Hartmann and Andreas Drexl. "Project scheduling with multiple modes: A comparison of exact algorithms". In: *Networks* 32.4 (1998), pp. 283–297. DOI: https://doi.org/10.1002/(SICI)1097-0037(199812)32:4<283::AID-NET5>3.0.CO;2-I.

[6] Kim van den Houten et al. *Proactive and Reactive Constraint Programming for Stochastic Project Scheduling with Maximal Time-Lags*. 2024. arXiv: 2409.09107 [cs.AI]. URL: https://arxiv.org/abs/2409.09107.

[7] Luke Hunsberger and Roberto Posenato. “Foundations of Dispatchability for Simple Temporal Networks with Uncertainty”. In: Jan. 2024, pp. 253–263. DOI: 10.5220/0012360000003636.

[8] Rainer Kolisch and Arno Sprecher. “PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program”. In: *European Journal of Operational Research* 96.1 (1997), pp. 205–216. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(96)00170-1. URL: https://www.sciencedirect.com/science/article/pii/S0377221796001701.

[9] Leon Lan and Joost Berkhout. *PyJobShop: Solving scheduling problems with constraint programming in Python*. 2025. arXiv: 2502.13483 [math.OC]. URL: https://arxiv.org/abs/2502.13483.

[10] Paul Morris. “Dynamic Controllability and Dispatchability Relationships”. In: *Integration of AI and OR Techniques in Constraint Programming: 11th International Conference, CPAIOR 2014, Cork, Ireland, May 19-23, 2014. Proceedings*. Vol. 8451. Lecture Notes in Computer Science. Springer, 2014, pp. 464–479. DOI: 10.1007/978-3-319-07046-9_33.

[11] Paul Morris, Nicola Muscettola, and Thierry Vidal. “Dynamic control of plans with temporal uncertainty”. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*. IJCAI’01. Seattle, WA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 494–499. ISBN: 1558608125.

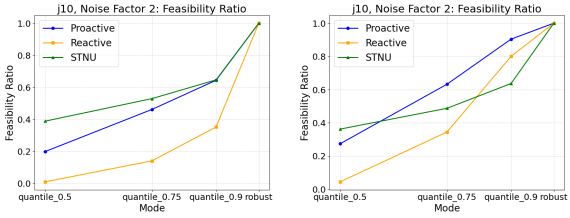
[12] Alfredo S. Ramos et al. “A Formulation for the Stochastic Multi-Mode Resource-Constrained Project Scheduling Problem Solved with a Multi-Start Iterated Local Search Metaheuristic”. In: *Mathematics* 11.2 (2023). ISSN: 2227-7390. DOI: 10.3390/math11020337. URL: https://www.mdpi.com/2227-7390/11/2/337.

A DelftBlue Parameters

The following configuration was used on the DelftBlue supercomputer to run all experiments:

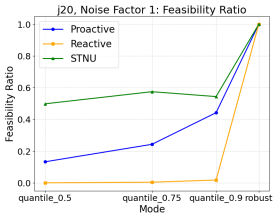
- **Job runtime limit:** 5 hours
- **CPUs per task:** 2
- **Memory per CPU:** 3968 MB
- **Number of tasks:** 1 (non-MPI job)
- **Partition used:** `compute`

B Results

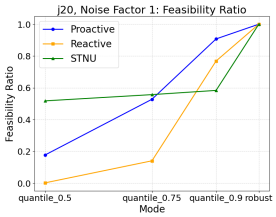


(a) Uniform distribution (b) Binomial distribution

Figure 5: Feasibility ratios across three approaches. Results are based on 100 instances with 10 tasks and a noise factor of 2.

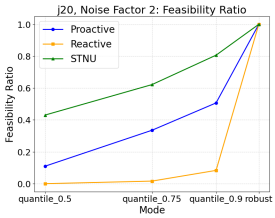


(a) Uniform distribution

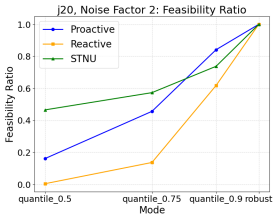


(b) Binomial distribution

Figure 6: Feasibility ratios across three approaches. Results are based on 100 instances with 20 tasks and a noise factor of 1.

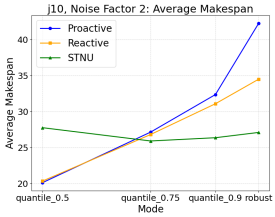


(a) Uniform distribution

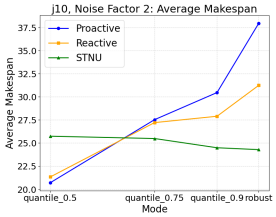


(b) Binomial distribution

Figure 7: Feasibility ratios across three approaches. Results are based on 100 instances with 20 tasks and a noise factor of 2.

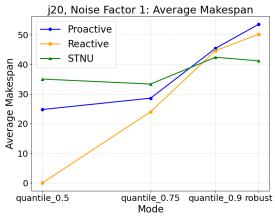


(a) Uniform distribution

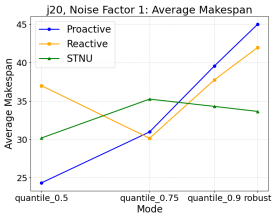


(b) Binomial distribution

Figure 8: Makespan values across three approaches. Results are based on 100 instances with 10 tasks and a noise factor of 2.

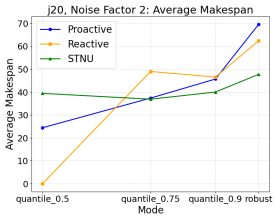


(a) Uniform distribution

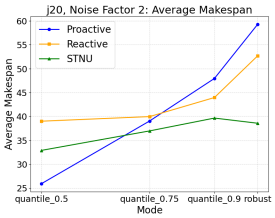


(b) Binomial distribution

Figure 9: Makespan values across three approaches. Results are based on 100 instances with 20 tasks and a noise factor of 1.

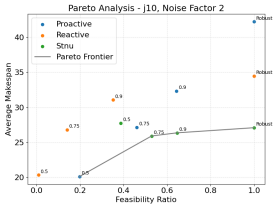


(a) Uniform distribution

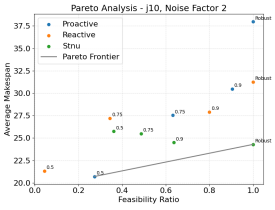


(b) Binomial distribution

Figure 10: Makespan values across three approaches. Results are based on 100 instances with 20 tasks and a noise factor of 2.

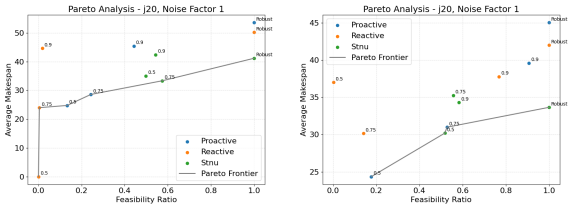


(a) Uniform distribution



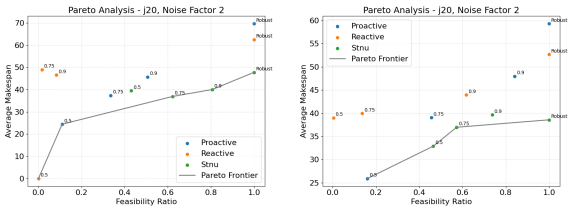
(b) Binomial distribution

Figure 11: Scatter plots of feasibility ratio against makespan for all sampling methods and both uniform and binomial distributions respectively. The grey line indicates the pareto optimal frontier.



(a) Uniform distribution (b) Binomial distribution

Figure 12: Scatter plots of feasibility ratio against makespan for all sampling methods and both uniform and binomial distributions respectively. The grey line indicates the pareto optimal frontier.



(a) Uniform distribution (b) Binomial distribution

Figure 13: Scatter plots of feasibility ratio against makespan for all sampling methods and both uniform and binomial distributions respectively. The grey line indicates the pareto optimal frontier.

Methods\Quantile	0.5	0.75	0.9	1
Reactive-Proactive	0.12	2.46e-47	3.88e-211	1.40e-267
Reactive-STNU	0.72	0.87	5.25e-13	3.57e-150
Proactive-STNU	3.28e-12	0.67	8.51e-35	1.12e-224

Table 2: p-values from Wilcoxon signed-rank tests comparing method pairs across different quantile-based sampling strategies. The significance level is set to $\alpha = 0.05$; values in red indicate non-significant results ($p > 0.05$).

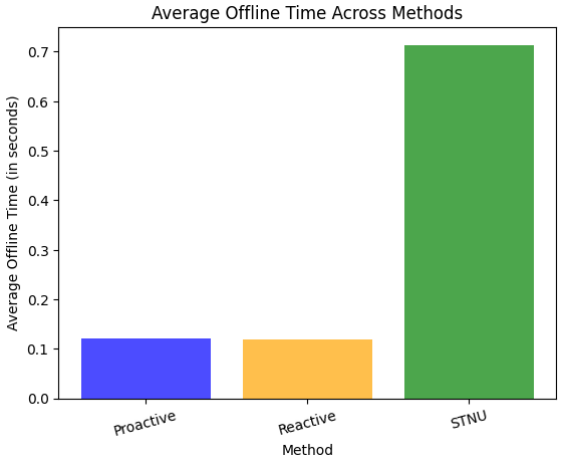


Figure 14: Average offline execution times across the three approaches. Results are based on 100 instances, each evaluated with 10 scenarios, under two noise factors (1 and 2) and two instance sizes (10 and 20).

Method	Instance	Noise	Avg Makespan
proactive_quantile_0.50	j10	1	19.9275
proactive_quantile_0.50	j10	2	21.3207
proactive_quantile_0.50	j20	1	24.6078
proactive_quantile_0.50	j20	2	26.2917
stnu	j10	1	22.2315
stnu	j10	2	23.2727
stnu	j20	1	31.7221
stnu	j20	2	35.8393

Table 3: Average makespan for proactive and STNU methods on job-shop instances under two noise levels using the binomial distribution. Values are rounded to 4 decimal places.

Method	Instance	Noise	Avg Makespan
proactive_quantile_0.50	j10	1	18.9432
proactive_quantile_0.50	j10	2	20.1532
proactive_quantile_0.50	j20	1	25.1153
proactive_quantile_0.50	j20	2	26.4150
stnu	j10	1	22.74444
stnu	j10	2	24.81136
stnu	j20	1	33.22820
stnu	j20	2	39.69285

Table 4: Average makespan for proactive and STNU methods on job-shop instances under two noise levels using the uniform distribution. Values are rounded to 4 decimal places.

C Supplementary Visualizations

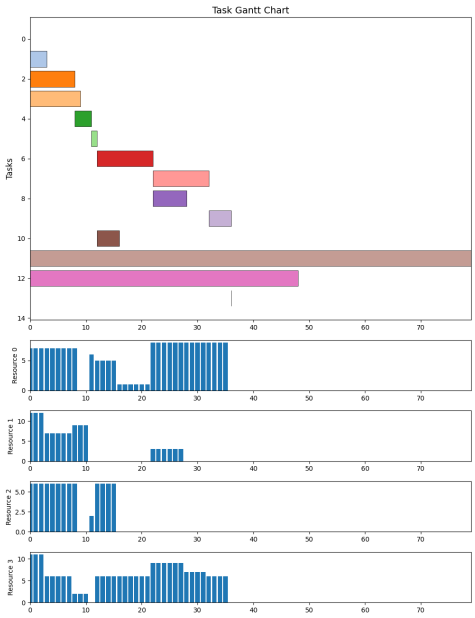


Figure 15: An example Gantt chart showing resource consumption and task scheduling for a specific instance. Tasks 11 and 12 represent deadline dummy tasks.