Annales Mathematicae et Informaticae
63 (2026) pp. 55–64
DOI: 10.33039/ami.2026.06.008
URL: https://ami.uni-eszterhazy.hu

Attribute-based Bloom filter tabu search for the Flexible Flowshop Problem

Levente Áron Fazekas, Károly Nehéz

Institute of Information Technology, University of Miskolc
{levente.fazekas,karoly.nehez}@uni-miskolc.hu

Abstract. In realistic flexible flowshop scheduling, candidate solutions are evaluated through detailed simulation, making optimization fundamentally evaluation-limited. In this regime, classical tabu search becomes increasingly inefficient: exact memory structures grow over time, yet fail to prevent near-duplicate evaluations that induce almost identical simulation trajectories.

This paper reframes tabu memory as a constant-time rejection mechanism rather than an exact history structure. We first replace exact tabu memory with Bloom filters, yielding a fixed-size, negligible-cost filtering layer. We then extend this idea to attribute-based tabu, where multiple Bloom filters operate on structural features of solutions rather than their full representations.

The resulting multi-filter mechanism enables early rejection of both revisits and structurally similar candidates before simulation. Experimental results show that exact Bloom tabu closely reproduces classical behavior, while attribute-based filtering reduces the number of costly evaluations by over 20% on average. The results highlight a central design principle: in simulation-dominated optimization, efficiency is achieved not by storing more history, but by making evaluation rare.

Keywords: flexible flowshop scheduling, tabu search, Bloom filter, probabilistic memory, simulation-based optimization

2020 *Mathematics Subject Classification:* Primary 90B35; Secondary 68W20, 68P10

1. Introduction

The Flexible Flowshop Scheduling Problem (FFSP) extends the classical flowshop by allowing multiple parallel machines at each processing stage [5, 16, 18]. Realistic

Submitted: March 31, 2026
Accepted: June 30, 2026
Published online: July 17, 2026

variants incorporate release times, due dates, sequence-dependent setup times, machine eligibility, and multiple performance criteria such as makespan or tardiness [6, 12, 14, 15, 17]. Recent work shows that flexible and hybrid flowshop scheduling remains an active research area, including online batching variants, flexible-shop neighborhood search, multi-objective evolutionary methods, energy-aware blocking models, and parallel/distributed tabu-search variants [8, 10, 11, 13, 19].

In such settings, evaluating a candidate solution is no longer trivial. Instead, it requires a discrete-event simulation (DES) that models the dynamic behavior of the system. As a result, optimization becomes evaluation-limited: the dominant cost lies not in generating candidates, but in assessing them.

This shift has important consequences. Classical tabu search relies on maintaining an exact memory of visited solutions. However, when evaluation dominates runtime, this exact memory becomes problematic: it grows over time, incurs increasing overhead, and still fails to prevent near-duplicate candidates that produce almost identical simulation outcomes.

This paper proposes a different perspective. Instead of treating tabu memory as an exact record of the past, we treat it as a rejection layer placed in front of the simulator. From this viewpoint, the goal is not perfect recall, but efficient filtering.

The contributions of this work are threefold:

- we reinterpret tabu memory as a constant-time rejection mechanism;
- we show that Bloom filters provide an efficient fixed-size implementation of this idea; and
- we extend the approach to attribute-based tabu, enabling similarity-aware filtering before simulation.

2. Discrete-event simulation model

In the FFSP, each job must pass through S stages, where each stage contains a set of parallel machines. The evaluation of a candidate schedule is performed by a discrete-event simulation that models the flow of jobs through the system.

As illustrated in Figure 1, jobs flow through buffers between stages. At each stage, two decisions must be made: which job to process next (dispatching), and which machine to assign (machine selection). These decisions determine the resulting schedule and its performance.

The objective values are computed from completion times C_j and due dates d_j:

$$C_{\max} = \max_j C_j, \quad T_j = \max(0, C_j - d_j),$$
$$T_{\max} = \max_j T_j, \quad \sum_j T_j, \quad \sum_j U_j.$$

Because simulation captures dynamic interactions such as machine contention and setup dependencies, evaluating even a single candidate can be computationally expensive.

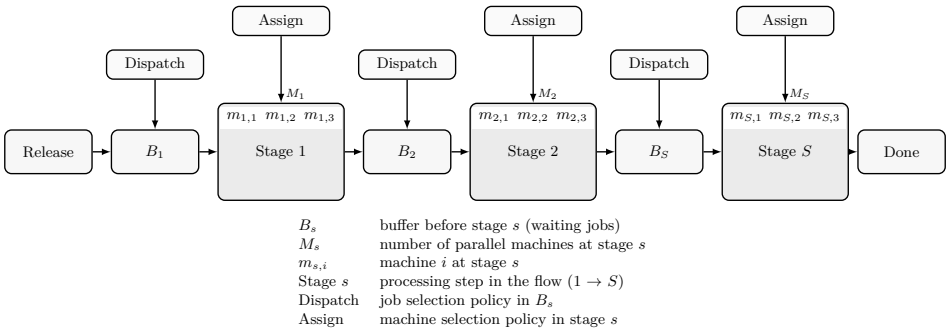


Figure 1. Discrete-event simulation model of the flexible flowshop. Jobs move through stage buffers, where dispatching and machine assignment decisions determine their progression.

3. Tabu search in the evaluation-limited regime

In a simulation-dominated setting, the role of tabu search must be reconsidered. This does not mean that tabu search itself is a weak baseline. On the contrary, tabu search is a long-established metaheuristic for combinatorial optimization and has repeatedly been compared with genetic algorithms, simulated annealing, iterated local search, GRASP, swarm-based methods, and other metaheuristics in scheduling and related optimization settings [2, 4, 7, 9, 15]. The aim of this paper is therefore more specific: instead of comparing tabu search as a whole against unrelated optimizers, we isolate the memory component of tabu search and ask whether its representation can be made cheaper and more informative in simulation-based scheduling.

Exact tabu memory prevents revisiting identical solutions, but this is often insufficient. The search space is vast, and the probability of generating exact duplicates is negligible. Instead, many distinct solutions produce nearly identical simulation trajectories, differing only in minor structural details.

From a computational perspective, these near-duplicates are the true source of inefficiency. They trigger expensive evaluations without contributing meaningful new information.

Thus, the fundamental question becomes: how can we prevent redundant evaluations, rather than exact revisits?

4. Bloom filters as tabu memory

Bloom filters provide a natural answer. They represent a set using a fixed-size bit array and multiple hash functions, enabling constant-time insertions and queries [1, 3].

Replacing exact tabu memory with a Bloom filter yields several advantages:

- constant-time operations independent of history size;
- fixed memory footprint;
- simple, cache-friendly implementation; and
- negligible overhead compared to simulation.

Bloom filters introduce false positives but no false negatives. In this context, a false positive simply skips a candidate early, which is often desirable.

The false-positive probability after n insertions is

$$p \approx \left(1 - e^{-kn/m}\right)^k,$$

with optimal $k \approx \frac{m}{n} \ln 2$.

For realistic parameter choices, even modest memory sizes yield extremely low error rates. More importantly, the filtering cost remains negligible compared to simulation.

This transforms tabu memory from a growing data structure into a lightweight rejection mechanism.

5. Attribute-based tabu filtering

The key extension of this work is to move beyond exact representations. Let $\phi_1(x), \ldots, \phi_q(x)$ denote structural features of a candidate solution, such as relative job orderings or stage-level patterns. Instead of storing full solutions, we maintain multiple Bloom filters over these attributes.

In the benchmark configuration, the multi-filter tabu memory consists of a single exact Bloom filter combined with a set of attribute-based filters. The exact component employs a 64-bit permutation hash and a Bloom filter of size 2^{15} with $k = 3$ hash probes.

The attribute layer uses the same Bloom filter parameters (2^{15} bits, $k = 3$) for each filter and encodes multiple structural views of the solution. These include stage-wise quantized processing-time sequence hashes, stage-wise position-bucketed processing-time histograms, an adjacent-pairs permutation signature, and a sampled inversion-order signature. For the five-stage instances considered, this results in a total of twelve attribute filters.

A candidate solution is rejected by the attribute layer if at least six of these filters indicate membership.

The tabu predicate becomes

$$\mathrm{Tabu}(x) = B_0(h(x)) \vee \big(B_1(\phi_1(x)) \wedge \cdots \wedge B_q(\phi_q(x))\big).$$

This formulation allows the search to reject not only exact revisits, but also structurally similar candidates. Importantly, all filtering occurs before simulation.

Conceptually, this represents a shift from representation-based memory to feature-based memory. Hashing becomes a mechanism for extracting structural signatures, and membership queries become similarity tests.

The effect is to reduce redundant evaluations, allowing computational effort to focus on genuinely new regions of the search space.

6. Why attribute-based filtering works

The effectiveness of attribute-based tabu filtering can be understood by examining the relationship between solution representations and their induced simulation trajectories.

Let x denote a candidate solution and let $\mathcal{S}(x)$ denote the outcome of its discrete-event simulation, including all relevant performance measures. In evaluation-limited settings, the optimization process operates not directly on x, but on $\mathcal{S}(x)$.

A key observation is that the mapping $x \mapsto \mathcal{S}(x)$ is many-to-one. Distinct candidate solutions often induce highly similar, or even identical, simulation trajectories. Formally, there exists an equivalence relation $\sim$ over the solution space such that

$$x \sim y \quad \Longrightarrow \quad \mathcal{S}(x) \approx \mathcal{S}(y),$$

where $\approx$ denotes equality or near-equality of objective values.

From this perspective, the search space is effectively partitioned into equivalence classes of solutions that behave similarly under simulation. Classical tabu search operates on exact representations and therefore treats all elements of an equivalence class as distinct, potentially exploring many representatives of the same class.

Attribute-based filtering provides a practical approximation of this equivalence structure. Let $\phi(x)$ denote a vector of structural features extracted from x. If the attribute mapping satisfies

$$\phi(x) = \phi(y) \quad \Longrightarrow \quad \mathcal{S}(x) \approx \mathcal{S}(y),$$

then ϕ acts as a coarse-grained representation of simulation behavior.

In this setting, tabu filtering over attributes does not merely prevent revisits in representation space, but instead suppresses redundant exploration of entire equivalence classes. The Bloom filters associated with attributes can therefore be interpreted as probabilistic caches over regions of the simulation space.

This interpretation clarifies both the strengths and limitations of the approach. When the attribute mapping aligns well with the structure of the simulation, redundant evaluations are effectively eliminated. However, if the mapping is too coarse, distinct high-quality solutions may be grouped together and prematurely excluded.

The central design problem is therefore to construct attribute mappings that preserve meaningful distinctions in simulation outcomes while collapsing irrelevant variation in representation space. In this sense, attribute-based tabu filtering can be viewed as a form of feature engineering for simulation-based optimization.

7. Experimental evaluation

The proposed methods were evaluated on a set of benchmark instances spanning multiple problem sizes and structural scenarios. Three tabu-memory variants were compared: classical exact memory, exact Bloom filtering, and attribute-based multi-filter tabu.

The results highlight a clear separation between implementation-level and policy-level effects. Exact Bloom filtering closely reproduces the behavior of classical tabu search, confirming that it can replace exact memory without loss of effectiveness while providing a fixed-size, constant-time alternative.

In contrast, the attribute-based approach alters the search dynamics. Across all tested instances, it consistently reduces the number of evaluated candidates, achieving an average reduction of 21.6%. This effect becomes more pronounced with increasing problem size, reflecting the growing prevalence of redundant evaluations in larger search spaces.

The impact on solution quality is more nuanced. In some cases, attribute filtering improves results by avoiding unproductive regions of the search space. In others, overly aggressive filtering excludes promising candidates. This behavior underscores the importance of attribute design and suggests the need for adaptive or problem-aware filtering strategies.

7.1. Evaluation-count reduction

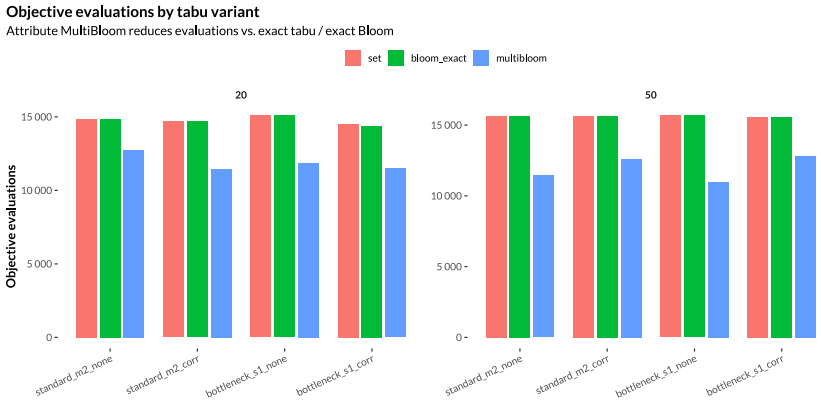
Table 1 reports the number of simulated candidates for each benchmark instance. The multi-filter attribute-based variant reduces the evaluation count in every comparison. The reduction ranges from 14.3% to 29.8%, with an overall average of 21.6%.

A clear scaling effect is observed: for $n = 20$, the average reduction is 19.6%, while for $n = 50$ it increases to 23.5%. This supports the interpretation that attribute-based filtering becomes increasingly effective as the density of simulation-equivalent candidates grows with problem size.

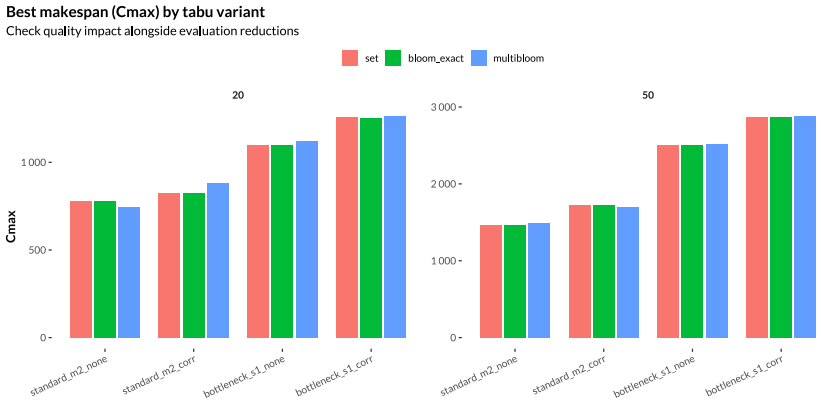
Table 1. Number of evaluated candidates. Reduction is computed relative to the classical set-based tabu memory.

n	Case	Set	Exact Bloom	Multi-Bloom	Reduction (%)
20	standard / no corr.	14866	14866	12745	14.3
20	standard / corr.	14684	14684	11472	21.9
20	bottleneck / no corr.	15089	15089	11850	21.5
20	bottleneck / corr.	14495	14361	11484	20.8
50	standard / no corr.	15661	15661	11438	27.0
50	standard / corr.	15664	15664	12610	19.5
50	bottleneck / no corr.	15671	15671	10995	29.8
50	bottleneck / corr.	15581	15581	12822	17.7

The same trend is visible in Figure 2, where the attribute-based method consistently lowers the number of expensive simulation calls. This reinforces the interpretation of the method as a pre-simulation filtering layer that removes redundant candidates before they incur computational cost.



(a) Evaluation count comparison.



(b) $C_{\max}$ comparison.

Figure 2. Benchmark plots illustrating evaluation counts and makespan across tabu-memory variants.

7.2. Solution-quality trade-off

The reduction in evaluation effort does not translate into uniform improvements in solution quality. Table 2 summarizes the relative change of the multi-filter method compared to the classical baseline, where negative values indicate improvement.

Two cases show clear improvements. For `standard_m2_none` with $n = 20$, the multi-filter method reduces evaluations by 14.3% while improving makespan

Table 2. Relative change of the multi-filter tabu memory against the classical set baseline. Entries marked *n/a* correspond to zero baseline tardiness values.

n	Case	$\Delta C_{\max}$	$\Delta \sum T$	$\Delta T_{\max}$	$\Delta \sum U$
20	standard / no corr.	−4.0%	n/a	n/a	+0
20	standard / corr.	+6.5%	+13.8%	+7.7%	+0
20	bottleneck / no corr.	+1.9%	n/a	n/a	+0
20	bottleneck / corr.	+0.6%	+0.5%	+2.1%	+0
50	standard / no corr.	+2.5%	+83.3%	+31.1%	+2
50	standard / corr.	−1.7%	−3.7%	−2.0%	+0
50	bottleneck / no corr.	+0.5%	+25.7%	+0.9%	+4
50	bottleneck / corr.	+0.6%	+1.2%	+0.6%	+0

by 4.0%. For `standard_m2_corr` with $n = 50$, it achieves a 19.5% reduction in evaluations while improving all reported time-based objectives.

In the remaining cases, however, the method exhibits a trade-off: fewer evaluations at the cost of slightly degraded performance. Across the eight matched instances, the multi-filter variant improves $C_{\max}$ in two cases and worsens it in six; it improves $\sum T$ in one case, worsens it in five, and leaves it unchanged in two.

These results reinforce the interpretation developed earlier. Exact Bloom filtering acts as an implementation-level improvement, preserving the behavior of classical tabu search while reducing overhead. In contrast, attribute-based multi-filter tabu modifies the effective search space by suppressing entire regions of simulation-equivalent solutions. Its benefit is most pronounced when evaluation cost dominates, but its impact on solution quality depends critically on how well the chosen attributes approximate meaningful equivalence classes in the simulation space.

This also clarifies the relationship between the proposed method and the broader tabu-search literature. Since tabu search has already been benchmarked extensively against other metaheuristics, the present comparison is deliberately orthogonal: it asks how much of the computational burden can be removed before the simulator is called. Exact Bloom tabu demonstrates that the classical tabu mechanism can be approximated by a fixed-size probabilistic memory, while attribute-based tabu shows that the same memory idea can be lifted from equality-based revisits to simulation-relevant structural similarity.

8. Discussion

The results reveal two distinct roles for Bloom filters in tabu search.

First, as an implementation tool, they provide a fixed-size, constant-time alternative to exact memory, eliminating bookkeeping overhead.

Second, as a modeling tool, they enable a new form of tabu logic based on structural similarity rather than exact equality.

In evaluation-limited optimization, the second role is particularly important. Reducing the number of simulations by even a modest percentage can translate into significantly deeper or broader exploration under a fixed time budget.

Future work should focus on adaptive attribute selection, dynamic filter management, and integration with aspiration criteria.

9. Conclusions

This paper proposed a reinterpretation of tabu search for simulation-based optimization.

Instead of maintaining exact history, tabu memory should act as a lightweight rejection layer that minimizes unnecessary evaluations. Bloom filters provide an efficient implementation of this idea, while attribute-based extensions enable similarity-aware filtering.

The experimental results demonstrate that this approach significantly reduces evaluation effort while preserving competitive solution quality.

The central insight is simple: in evaluation-dominated optimization, efficiency comes not from remembering everything, but from evaluating less.

References

[1] P. S. Almeida, C. Baquero, N. Preguiça, D. Hutchison: *Scalable Bloom Filters*, Information Processing Letters 101.6 (2007), pp. 255–261, issn: 0020-0190, doi: 10.1016/j.ipl.2006.10.007, url: https://www.sciencedirect.com/science/article/pii/S0020019006003127.

[2] K. A. G. Araújo, E. G. Birgin, D. P. Ronconi: *Local search and trajectory metaheuristics for the flexible job shop scheduling problem with sequencing flexibility and position-based learning effect*, 2024, arXiv: 2403.16787 [math.OC], url: https://arxiv.org/abs/2403.16787.

[3] B. H. Bloom: *Space/time trade-offs in hash coding with allowable errors*, Commun. ACM 13.7 (July 1970), pp. 422–426, issn: 0001-0782, doi: 10.1145/362686.362692.

[4] A. Bouketta: *Hybrid Metaheuristic Combining the Dragonfly Algorithm and Tabu Search for the Traveling Salesman Problem*, 2026, arXiv: 2606.09529 [cs.NE], url: https://arxiv.org/abs/2606.09529.

[5] P. Brucker: *Scheduling algorithms*, 4th ed., Springer Berlin, Heidelberg, 1999, pp. XII, 367, isbn: 978-3-540-24804-0, doi: 10.1007/978-3-540-24804-0.

[6] L. A. Fazekas, K. Nehez: *Multi-objective genetic and memetic algorithms in flexible flow-shop scheduling*, in: Annales Mathematicae et Informaticae, vol. 61, 2025, pp. 94–107, doi: 10.33039/ami.2025.10.011.

[7] F. Glover: *Tabu search: A tutorial*, Interfaces 20.4 (1990), pp. 74–94, doi: 10.1287/inte.20.4.74.

[8] C. Hertrich, C. Weiss, H. Ackermann, S. Heydrich, S. O. Krumke: *Online Algorithms to Schedule a Proportionate Flexible Flow Shop of Batching Machines*, 2020, arXiv: 2005.03552 [cs.DS], url: https://arxiv.org/abs/2005.03552.

[9] E. Hosseini: *Tabu Search and Simulated Annealing metaheuristic algorithms applied to the RoRo vessel stowage problem*, 2023, arXiv: 2301.04468 [cs.AI], url: https://arxiv.org/abs/2301.04468.

[10] A. Janiak, D. Kowalczyk, M. Lichtenstein: *Parallel/Distributed Tabu Search for Scheduling Microprocessor Tasks in Hybrid Flowshop*, 2025, arXiv: 2509.11396 [cs.DC], url: https://arxiv.org/abs/2509.11396.

[11] F. Liu, X. Li, C. Lu, W. Gong: *Adaptive Knowledge-based Multi-Objective Evolutionary Algorithm for Hybrid Flow Shop Scheduling Problems with Multiple Parallel Batch Processing Stages*, 2024, arXiv: 2409.18524 [cs.NE], url: https://arxiv.org/abs/2409.18524.

[12] K. Mihaly, G. Kulcsar: *A new many-objective hybrid method to solve scheduling problems*, International Journal of Industrial Engineering and Management 14.4 (2023), pp. 326–335, doi: 10.24867/IJIEM-2023-4-342.

[13] A. Missaoui, C. Ozturk, B. O'Sullivan: *Refined Iterated Pareto Greedy for Energy-aware Hybrid Flowshop Scheduling with Blocking Constraints*, 2025, arXiv: 2510.03377 [cs.AI], url: https://arxiv.org/abs/2510.03377.

[14] B. Naderi, M. Zandieh, V. Roshanaei: *Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness*, The International Journal of Advanced Manufacturing Technology 41 (2009), pp. 1186–1198, doi: 10.1007/s00170-008-1569-3.

[15] C. Oğuz, M. F. Ercan: *A Genetic Algorithm for Hybrid Flow-shop Scheduling with Multiprocessor Tasks*, Journal of Scheduling 8.4 (July 2005), pp. 323–351, issn: 1099-1425, doi: 10.1007/s10951-005-1640-y.

[16] M. L. Pinedo: *Scheduling, Theory, Algorithms, and Systems*, 6th ed., Springer Cham, 2012, pp. XVII, 698, isbn: 978-3-031-05920-9, doi: 10.1007/978-3-031-05921-6.

[17] I. Ribas, R. Leisten, J. M. Framiñan: *Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective*, Computers & Operations Research 37.8 (2010), Operations Research and Data Mining in Biological Systems, pp. 1439–1454, issn: 0305-0548, doi: 10.1016/j.cor.2009.11.001, url: https://www.sciencedirect.com/science/article/pii/S0305054809002883.

[18] R. Ruiz, J. A. Vázquez-Rodríguez: *The hybrid flow shop scheduling problem*, European Journal of Operational Research 205.1 (2010), pp. 1–18, issn: 0377-2217, doi: 10.1016/j.ejor.2009.09.024, url: https://www.sciencedirect.com/science/article/pii/S0377221709006390.

[19] J. C. Seck-Tuoh-Mora, N. J. Escamilla-Serna, J. Medina-Marin, N. Hernandez-Romero, I. Barragan-Vite, J. R. Corona-Armenta: *A global-local neighborhood search algorithm and tabu search for flexible job shop scheduling problem*, 2020, arXiv: 2010.12702 [cs.AI], url: https://arxiv.org/abs/2010.12702.