Large Neighborhood Search for Generalized Parallel Machine Scheduling

Lukas Frühwirth, Nysret Musliu, and Felix Winter

DBAI, TU Wien, Vienna, Austria
{lukas.fruehwirth, nysret.musliu, felix.winter}@tuwien.ac.at

Keywords: Large Neighborhood Search · Parallel Machine Scheduling · Constraint Programming · Rich Scheduling · Production Scheduling

1 Introduction

Real-world production scheduling problems often combine constraints from multiple classical formulations such as Unrelated Parallel Machine Scheduling, Flexible Job Shop Scheduling, and Resource-Constrained Project Scheduling (RCPSP). We recently introduced the Generalized Parallel Machine Scheduling (GPMS) framework [1], which assigns jobs to unit-capacity or non-capacitated machines, minimizing a weighted sum of objectives such as tardiness, makespan, and setup time. The framework unifies machine eligibility, sequence- and machine-dependent setup times, machine calendars, precedence relations with time lags, and secondary resources with time-varying capacities and pulse and step demands. A complete formal definition is available in [1].

While the proposed constraint programming (CP) models solve small-to-medium GPMS instances effectively, scalability remains challenging: instances with more than 20 jobs frequently cannot be solved to optimality within practical time limits, and complex constraints such as secondary resources with step demands increase model size significantly, limiting improvements over heuristic solutions even on smaller instances.

This scaling limitation motivates the development of a metaheuristic approach. Large Neighborhood Search (LNS) iteratively destroys and repairs parts of a solution, exploring large neighborhoods that would be intractable for classical local search [7, 6]. LNS with CP-based repair has proven particularly effective in the scheduling domain, achieving state-of-the-art results for single-mode scheduling [3], RCPSP variants [8], and flexible job shop scheduling [2, 4]. However, the specific operator design for the full GPMS constraint set remains an open challenge.

In this work, we propose an LNS framework for GPMS with domain-specific neighborhoods, CP-based and heuristic repair operators, and various search strategies. We present preliminary experimental results indicating the effectiveness of the approach.

Algorithm 1 LNS for GPMS

1: $\sigma \leftarrow \sigma_0$
2: **while** elapsed time $< t_{\max}$ **do**
3: $D \leftarrow$ select a subset of jobs to destroy (neighborhood $\mathcal{N}$, strategy $\mathcal{S}$)
4: Construct sub-instance $\mathcal{I}'$ from D and σ
5: $\sigma'_D \leftarrow \mathcal{R}(\mathcal{I}', t_{\text{repair}})$
6: **if** σ'_D feasible $\wedge$ obj(Merge$(\sigma, \sigma'_D)) \leq$ obj(σ) **then**
7: $\sigma \leftarrow$ Merge(σ, σ'_D);
8: **return** σ

2 Large Neighborhood Search Framework

Our LNS approach iteratively improves an initial feasible schedule produced by the randomized construction heuristic (CH) from [1]. Algorithm 1 outlines the core loop: σ_0 is the initial heuristic schedule, σ the current best schedule, and $\mathcal{R}$ the repair operator with time budget t_{repair}. In each iteration, a neighborhood operator selects a subset of jobs D to destroy. A repair operator re-optimizes their assignment and timing, and the result is merged back if it does not worsen the objective (greedy acceptance). The sub-problem can be formulated either by extracting a self-contained sub-instance or by fixing variables in the full CP model. Each run uses a fixed configuration of neighborhood, repair operator, and search strategy; adaptive operator selection is planned for future work.

Neighborhoods. All neighborhoods share a parameter $\bar{n}$ controlling the maximum number of destroyed jobs. The *machine* neighborhood iteratively adds machines (in an order determined by the search strategy) and all their assigned jobs to D until $|D| \geq \bar{n}$, always including at least one machine. The *time-window* neighborhood selects a reference time t based on the search strategy and alternately adds the closest jobs to the left and right of t (by setup start time) until $|D| = \bar{n}$. The *sequence* neighborhood destroys a contiguous block of up to $\bar{n}$ jobs on a single machine, targeting fine-grained resequencing. The *random* neighborhood uniformly samples $\bar{n}$ jobs, serving as a diversification baseline.

Sub-Instance Construction. The destroyed jobs and their affected machines are extracted into a self-contained GPMS sub-instance. Boundary conditions, such as sequence-dependent setups, machine calendars, secondary-resource capacities, and precedence relations, are encoded into the sub-instance so that any feasible sub-solution can be merged back into the global schedule.

As an alternative, a *variable-fixing* strategy fixes all jobs outside D in the full CP model rather than extracting a sub-instance. This avoids the complexity of boundary handling but incurs higher presolving overhead.

Repair Operators. Each repair operator works within a time budget t_{repair}. The *CP-SAT* repair uses the interval-variable model from [1], solved with Google's CP-SAT solver [5], warm-started with the current sub-schedule. It can operate

on either an extracted sub-instance or on the full model with variable fixing. The *construction heuristic* repair applies the randomized heuristic from [1] with top-k selection for diversification and only operates on the extracted sub-instance.

Search Strategies. A *random* strategy selects neighborhood parameters uniformly at random. A *sliding* strategy systematically traverses the schedule, cycling through machines or partitioning jobs into temporal groups of size $\bar{n}$. An *objective-guided* strategy prioritizes schedule regions with high penalty contributions, e.g., selecting machines with the highest tardiness or time windows containing the costliest setups.

3 Preliminary Results

Experimental Setup. All experiments were run with $t_{\max} = 600\,\text{s}$ and $t_{\text{repair}} = 30\,\text{s}$. The maximum destroy size was set to $\bar{n} = 40$ for instances 1–5 and $\bar{n} = 10$ for instances 6–10.

Instances. We evaluated on 10 GPMS benchmark instances from [1]. Instances 1–5 are real-world instances: 1–3 are pure unrelated parallel machine problems with sequence-dependent setups, and 4–5 include precedence relations and secondary resources. Instances 6–10 are generated instances with 20 to 50 jobs, featuring complex precedence graphs with minimum/maximum time lags and conditional eligibilities, as well as secondary resources with pulse and step demands.

Evaluated Configurations. Table 1 reports results for eight configurations. Since CH from [1] is currently the best-performing heuristic for large and heavily constrained GPMS instances, we report all objective values relative to the CH solution. IV_w represents the interval-variable CP-SAT model warm-started with CH. The seven LNS variants are labeled N-P-S, where N denotes the neighborhood (RD = random, M = machine, T = time-window), P the sub-problem strategy (VF = variable fixing, SI = sub-instance extraction), and S the search strategy (R = random, S = sliding). All LNS variants use the warm-started IV_w model as repair operator.

Discussion. These results are preliminary, and we draw no firm conclusions from them at this stage. The overall tendency is that LNS improves over the standalone IV_w model, with the gap widening on larger instances. On instances 3–5 ($|J| \geq 1167$), IV_w and all variable-fixing variants fail to improve over CH, while sub-instance-based configurations reduce the objective to as low as 0.285 of the baseline. On the small generated instances (6–7), IV_w already performs well, finding the optimum on instance 6, and several LNS variants match its results. On the larger generated instances (8–10), IV_w fails to improve over CH, whereas LNS variants achieve improvements of up to 15%. No single configuration dominates, however: the best results are spread across different neighborhoods and search strategies. This indicates that LNS is a promising direction for GPMS,

Table 1. Objective values relative to the CH baseline ($t_{\max} = 600\,\mathrm{s}$). Values below 1 indicate improvement over CH; lower is better. Best per instance in bold.

Inst.	$\lvert J\rvert$	IV_w	RD-VF-R	M-VF-R	M-SI-R	M-SI-S	T-VF-R	T-SI-R	T-SI-S
1	72	0.540	0.567	0.554	—	—	**0.538**	0.543	0.561
2	601	0.977	0.969	0.899	—	—	0.755	0.619	**0.499**
3	1167	1.000	1.000	1.000	0.408	**0.285**	1.000	0.764	0.697
4	1694	1.000	1.000	1.000	1.000	1.000	1.000	**0.959**	0.962
5	2142	1.000	1.000	1.000	1.000	1.000	1.000	0.989	**0.959**
6	20	**0.864**	**0.864**	**0.864**	**0.864**	**0.864**	0.910	0.910	0.922
7	20	0.835	0.835	**0.834**	**0.834**	0.838	0.835	0.838	0.957
8	50	1.000	1.000	1.000	0.979	0.978	1.000	**0.960**	0.980
9	50	1.000	0.987	1.000	0.996	0.996	**0.980**	0.981	0.984
10	50	1.000	0.980	1.000	**0.852**	**0.852**	0.989	1.000	0.978

but that no fixed variant can yet be regarded as generally preferable, which motivates the adaptive operator selection we leave for future work. With only ten instances, these observations remain indicative rather than conclusive, and a reliable assessment will require the substantially larger and more diverse instance set we plan to evaluate in the extended version of this study.

References

1. Frühwirth, L., Einspieler, C., Musliu, N., Winter, F.: GPMS: A generalized parallel machine scheduling framework with rich temporal and resource constraints. In: Proceedings of the 36th International Conference on Automated Planning and Scheduling (ICAPS) (2026), preprint: https://tucloud.tuwien.ac.at/index.php/s/8ZqHFkt75bPFLDc
2. Kasapidis, G.A., Paraskevopoulos, D.C., Mourtos, I., Repoussis, P.P.: A unified solution framework for flexible job shop scheduling problems with multiple resource constraints. European Journal of Operational Research **320**(3), 479–495 (2025). https://doi.org/10.1016/j.ejor.2024.08.010
3. Laborie, P., Godard, D.: Self-adapting large neighborhood search: Application to single-mode scheduling problems. In: Proceedings of the 3rd Multidisciplinary International Scheduling Conference (MISTA). pp. 276–284 (2007)
4. Pacino, D., Van Hentenryck, P.: Large neighborhood search and adaptive randomized decompositions for flexible jobshop scheduling. In: Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Three. p. 1997–2002. IJCAI'11, AAAI Press (2011)
5. Perron, L., Didier, F., Gay, S.: The CP-SAT-LP solver. In: 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 280, pp. 3:1–3:2. Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPIcs.CP.2023.3
6. Pisinger, D., Ropke, S.: Large neighborhood search. In: Gendreau, M., Potvin, J.Y. (eds.) Handbook of Metaheuristics, pp. 99–127. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-319-91086-4˙4

7. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Principles and Practice of Constraint Programming (CP 1998). LNCS, vol. 1520, pp. 417–431. Springer (1998). https://doi.org/10.1007/3-540-49481-2˙30
8. Vilím, P., Laborie, P., Shaw, P.: Failure-directed search for constraint-based scheduling. In: Michel, L. (ed.) Integration of AI and OR Techniques in Constraint Programming. pp. 437–453. Springer International Publishing, Cham (2015)