Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Conflict-Weighting Variable Selection in Reactive Multi-Objective Job-Shop and RCPSP

Lillia Ouali
University of Bejaia
Madani Bezoui
mbezoui@cesi.fr

LINEACT, CESI
Kamal Amroun
University of Bejaia

Research Article

Keywords:

Posted Date: November 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10008411/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Conflict-Weighting Variable Selection in Reactive Multi-Objective Job-Shop and RCPSP

Lillia Ouali[1], Madani Bezoui[2*], Kamal Amroun[1]

[1]Laboratory of Medical Informatics and Intelligent and Dynamic Environments (LIMED), University of Bejaia, Bejaia, Algeria.
[2]LINEACT, CESI, Nancy, France.

*Corresponding author(s). E-mail(s): mbezoui@cesi.fr;
Contributing authors: lillia.ouali@univ-bejaia.dz;
kamal.amroun@univ-bejaia.dz;

Abstract

Conflict-weighting variable selection (`dom/wdeg` and its descendants) is central to constraint-programming (CP) search for scheduling, yet its variants are easily conflated, and their behaviour under repeated multi-objective re-solving of perturbed snapshots with long-scope propagators (`noOverlap`, `cumulative`) had not been characterised. We first make the weight-increment rules explicit: crediting a propagator failure to the entire scope of the failing constraint is the original `dom/wdeg` rule of 2004, as implemented by Choco's `domOverWDegSearch`; per-variable attribution is a later refinement. We compare scope-sharing, per-variable weighting, and a History of Decisions (HD) augmentation in a single Choco harness where only the increment rule varies, on **162** standard job-shop and PSPLIB Resource-Constrained Project Scheduling Problem (RCPSP) instances with paired seeds and five reactive snapshots (**159** form the test set). Three findings emerge: scope-sharing coincides with Choco `dom/wdeg` up to tie-breaking; measured instantaneous scope-freshness is low, making the sufficient high-freshness convergence bound vacuous on these benchmarks, and direct search-tree measurements show the two rules are not interchangeable in the tested regime; and HD leaves hypervolume unchanged while reducing search effort on the RCPSP instances where proofs close. Against contextual priority-rule Non-dominated Sorting Genetic Algorithm (NSGA-II/III) baselines at an equal per-snapshot budget, the CP selectors attain higher hypervolume and far lower internal solver time on RCPSP only; NSGA leads on job-shop hypervolume, a gap that persists when its fronts are capped to the CP front cardinality. Conflict-weighting gains are thus family-dependent; we provide the controlled, reproducible benchmark that delimits where the variants genuinely differ.

Keywords: Constraint programming, Variable-ordering heuristics, dom/wdeg, Reactive scheduling, Multi-objective optimisation, RCPSP

1 Introduction

Manufacturing environments are increasingly characterised by volatility: machine breakdowns, rush orders, and processing-time variations routinely invalidate static schedules (Vieira et al. 2003; Ouelhadj and Petrovic 2009), and the inability to react swiftly leads to bottlenecks, missed deadlines, and amplified production costs.

While data-driven approaches such as Deep Reinforcement Learning (DRL) have surged for reactive scheduling (Xu et al. 2025), Constraint Programming (CP) remains a strong and widely used exact framework for scheduling problems with rich constraints (Laborie et al. 2018; Lan and Berkhout 2025), relying on systematic tree search guided by variable- and value-selection heuristics. The most widely used generic variable selector remains `dom/wdeg` (Boussemart et al. 2004), which maintains a scalar failure weight per constraint and scores a variable by the sum of the weights of the constraints in whose scope it appears, divided by its current domain size, so that the full weight of a failing constraint is credited to *every* variable in its scope. A line of refinements distributes that weight more finely: per-variable attribution (Wattez et al. 2019), explanation-based credit (Hébrard and Siala 2017), and recency-weighted conflict history (Habet and Terrioux 2019). For scheduling, Grimes and Hébrard (2015) showed that conflict-directed search on a disjunctive decomposition is highly effective on job-shop variants. What is far less understood is how these weighting schemes behave under *repeated multi-objective re-solving*, where the same instance is re-solved across perturbed snapshots, front quality matters as much as time to first solution, and the propagators (`noOverlap`, `cumulative`) have unusually long scopes, so that the choice between crediting the whole scope (2004) and per-variable, future-scope attribution (2019) could plausibly matter.

Rather than propose a new heuristic, we study how the established conflict-weighting schemes behave in exactly this setting, organised around one explicit research question: *When scheduling propagators have long scopes, does the original* `dom/wdeg` *scope-sharing rule behave like per-variable conflict attribution, or do the two produce genuinely different search behaviour under reactive multi-objective re-solving, and does a decision-driven augmentation (HD) change front quality, or only search effort?*

Our starting point is a clarification with practical consequences: the "scope-based" attribution that one might be tempted to present as a novel generalisation of `dom/wdeg` is, algebraically, the *original* 2004 weighted degree (Boussemart et al. 2004); it is the per-variable attribution that is the later development (Wattez et al. 2019). We therefore implement three increment rules, 2004 scope-sharing, per-variable refinement, and a branching-history augmentation, in a single Choco harness in which the increment rule is the only difference, and compare them with search-sensitive measures (nodes, failures, proof closure) alongside hypervolume, in an ε-constraint multi-objective solver run over reactive snapshots.

Specifically, our contributions are:

(i) **A clarification with a testable prediction.** We give the precise algebra (Section 4.2) showing that scope-based conflict attribution coincides with `dom/wdeg2004` and with Choco's `domOverWDegSearch`, and we prove (Proposition 1) that the scope-sharing and per-variable weightings differ by a relative factor bounded by the *scope-freshness*, the fraction of a constraint's failures at which its scope variables are still unbound. This yields a falsifiable prediction: the two rules converge when scope-freshness is high. We instrument the solver to measure this quantity and find it low on our benchmarks, making the sufficient high-freshness convergence bound vacuous (Sections 4.3 and 6.4); direct search-tree measurements then show that the two rules are not interchangeable in the tested regime.

(ii) **A controlled comparison** of the three increment rules on long-scope scheduling propagators, in a repeated multi-objective ε-constraint setting not previously examined for these schemes, using a paired-seed design and branching-sensitive statistics (nodes, failures, closure rate, measured scope-freshness), not only hypervolume under a fixed budget.

(iii) **Front-quality findings** on 162 standard instances (82 job-shop; 80 PSPLIB j30/j60 RCPSP), 5 paired seeds and 5 snapshots; Hypervolume (HV) and efficiency statistics use the 159-instance test set (the three job-shop tuning instances excluded): on a family-comparable normalised hypervolume, Choco `dom/wdeg` and our pure `dom/wdeg2004` coincide up to tie-breaking, both clearly beat `activity` under the shared no-restart protocol, and their advantage over `dom` is weak and family-dependent.

(iv) **Search-effort findings.** The controlled search-tree experiment separates what hypervolume cannot: the per-variable refinement yields different weighting trajectories and descriptive node/failure profiles (its node-count differences are not uniformly statistically significant), while the branching-history augmentation, the only component that algorithmically departs from `dom/wdeg2004`, leaves hypervolume unchanged but reduces the nodes explored on the RCPSP instances where proofs close.

(v) **A contextual CP-metaheuristic comparison.** Under an equal per-snapshot budget, the CP selectors use one-to-two orders of magnitude less internal solver time than the priority-rule NSGA-II/NSGA-III baselines on RCPSP (with higher hypervolume there), while on job-shop NSGA attains higher hypervolume at a comparable budget; CP retains anytime feasibility, with optimality guarantees for the ε-subproblems that close within the time limit.

Throughout this paper, "reactive" denotes repeated re-optimisation of perturbed snapshots under controlled perturbations: conflict weights are reset at each snapshot, and no cross-snapshot learning or weight transfer is performed. The HD term is deliberately treated as a *minor diagnostic augmentation*, the one component that algorithmically departs from `dom/wdeg2004`, not as the contribution of the paper.

The increment rule is consequential in scheduling specifically because global propagators concentrate many activities in few constraints: a single `noOverlap` on a machine, or `cumulative` on a resource, carries most of the model's failure signal, so how that

signal is divided among the activities in its scope is precisely the design decision that separates the weighting variants. Repeated re-solving sharpens the issue: weights are relearned from scratch at every snapshot, so a rule that shapes the early tree differently changes both the returned approximation and the chance of proof closure within budget.

2 Literature Review

2.1 Evolution from Static to Dynamic and Reactive Scheduling

Classical scheduling literature historically focused on the deterministic Job-Shop Scheduling Problem (JSSP), famously formalised by Muth and Thompson (1963). Modern manufacturing, however, faces constant disturbances such as breakdowns and rush orders, and research has consequently pivoted toward dynamic and reactive scheduling: Vieira et al. (2003) give the canonical framework of rescheduling strategies, policies, and methods, and Ouelhadj and Petrovic (2009) survey dynamic scheduling in manufacturing systems; for project scheduling, Herroelen and Leus (2005) survey the corresponding literature on scheduling under uncertainty. Reactive approaches focus on rapidly repairing the schedule upon the detection of a disruption (Sáez Bustos and Parada 2025; Maenhout and Vanhoucke 2024); rescheduling for newly arriving orders under disruption constraints remains an active line of work (Rener et al. 2025). However, these traditional reactive methods often rely on simple, myopic dispatching rules that sacrifice global optimality for speed. A critical gap remains in developing frameworks that match the speed of dispatching rules while preserving the rigorous bounds of exact optimisation.

2.2 Deep Reinforcement Learning in Reactive Scheduling

Deep Reinforcement Learning has recently been used for reactive scheduling because it produces fast dispatching decisions, typically by treating the problem as a Markov Decision Process (Duan et al. 2025; Wu et al. 2025; Xu et al. 2025; Bezoui et al. 2023a); such methods, however, generally lack exact feasibility and optimality guarantees (Pedrosa and Puig 2023). Since this paper focuses on complete CP search, DRL is a contrasting family of approaches rather than the methodological baseline, and we do not review it further.

2.3 Constraint Programming and Conflict-Directed Weighting

Constraint Programming is a well-established exact framework for scheduling problems with rich constraints (Laborie et al. 2018; Da Col and Teppan 2022; Teppan 2022; Lan and Berkhout 2025; Geibinger et al. 2024), and at its algorithmic core lies the variable-ordering heuristic. The dominant generic family is conflict-directed weighting, introduced as `dom/wdeg` by Boussemart et al. (2004): each constraint accumulates a scalar weight on every domain wipeout it causes, and a variable is ranked by the sum of the weights of the constraints in its scope divided by its domain size. Subsequent work refines *how* the credit is assigned and decayed rather than overturning the idea: Wattez et al. (2019) move from a per-constraint scalar to a per-variable weight; Hébrard and

Siala (2017) credit only the variables an explanation shows responsible for the failure; Habet and Terrioux (2019) replace the cumulative count with a recency-weighted average of conflicts (CHS). These distinctions are subtle and easily conflated, yet they define different heuristics; making the precise increment rule explicit (Section 4.2) is a prerequisite for any claim of novelty. For scheduling specifically, Grimes and Hébrard (2015) demonstrated that conflict-directed search over a disjunctive decomposition, combined with restarts and solution-guided branching, is among the most effective complete methods on job-shop variants. The missing point is not whether conflict-weighting works, it does, but how its increment rules behave under repeated multi-objective re-solving with the long-scope propagators typical of scheduling (Chang et al. 2022; Pleier and Schilp 2024; Bär 2022).

2.4 Gap Analysis and Contribution

While much recent multi-objective scheduling research targets the Flexible Job-Shop Problem (Aribi et al. 2023; Ebufegha and Li 2022; Bezoui et al. 2023b), we deliberately restrict the model to the classical JSSP and to RCPSP, isolating the branching rule, the object of our study, from the confounding effect of routing decisions. Two gaps then motivate this study. First, the precise relationship between the weighting variants is often left implicit, and the original 2004 scope-sharing rule (Boussemart et al. 2004) is easily mistaken for a novel "scope-based" generalisation, when in fact the per-variable attribution (Wattez et al. 2019), the explanation-based credit (Hébrard and Siala 2017), and the conflict-history weighting (Habet and Terrioux 2019) are the refinements that came *after* it; Section 4.2 states the algebra explicitly so that the comparison rests on the actual increment rules. Second, while conflict-directed search is known to be strong on static job-shop (Grimes and Hébrard 2015), its behaviour under repeated multi-objective ε-constraint re-solving of perturbed snapshots, with long-scope `noOverlap`/`cumulative` propagators and front quality as the target, has not, to our knowledge, been characterised. Our contribution is a controlled comparison of the three increment rules in exactly that setting, with a precise account of which differences affect front quality and which affect search effort.

3 Problem Formulation

3.1 Notation and Variables

The multi-objective job-shop scheduling model solved at each reactive snapshot is formalised using the notation in Table 1.

3.2 Mathematical Model

The core constraints governing the schedule are formalised as follows:

1. **Precedence Constraints:**

$$S_{i,j+1} \geq S_{i,j} + p_{i,j} \quad \forall i,\ j \in \{1, \ldots, |J_i| - 1\} \tag{1}$$

Table 1 Mathematical Notation.

Symbol	Description
$\mathcal{J}, \mathcal{M}$	Set of jobs $J_1 \dots J_n$ and machines $M_1 \dots M_m$
$O_{i,j}$	The j-th operation of job J_i
$M_{i,j}$	Deterministic machine assigned to $O_{i,j}$
$p_{i,j}$	Processing time of $O_{i,j}$
r_i	Release date of job J_i (job arrivals)
$S_{i,j} \in \mathbb{N}$	Decision variable: Start time of $O_{i,j}$
$C_{i,j}$	Completion time of $O_{i,j}$ $(S_{i,j} + p_{i,j})$

2. **Disjunctive (No-Overlap) Constraints:**

$$S_{i,j} \geq C_{i',j'} \quad \vee \quad S_{i',j'} \geq C_{i,j} \quad \forall (O_{i,j}, O_{i',j'}) \text{ where } M_{i,j} = M_{i',j'} = M_k \tag{2}$$

3. **Release Dates:**

$$S_{i,j} \geq r_i \quad \forall i, j \tag{3}$$

All start variables take integer domains $S_{i,j} \in [0, H]$, where the horizon H is the sum of all processing times in the snapshot, a valid upper bound on any active schedule, and release dates are enforced by constraint (3); the RCPSP variables of Section 3.3 use the same sum-of-durations horizon.

We note that the implemented breakdown model is a *degradation* model, not a calendar-unavailability model: a breakdown on machine M_k is realised by inflating the processing times of the operations on M_k (Section 3.5), rather than by forbidding processing during an unavailable window $[a_k, b_k]$. The two are different; we use the degradation form throughout and do not impose any calendar-window constraint.

3.3 RCPSP Mathematical Model

For the RCPSP family, each activity $i \in \mathcal{V} = \{0, 1, \dots, n, n+1\}$ has a start-time variable $S_i \in \mathbb{N}$, a duration d_i, a completion time $C_i = S_i + d_i$, and a release date r_i. Activities 0 and $n+1$ are the dummy source and sink, with zero duration; they are used to preserve the precedence graph but are excluded from the flowtime objective. Renewable resources are indexed by $k \in \mathcal{R}$, with capacities R_k and per-activity demands q_{ik}. The precedence set is denoted by $\mathcal{P} \subseteq \mathcal{V} \times \mathcal{V}$.

The RCPSP constraints solved at each snapshot are

$$S_j \geq S_i + d_i \qquad \forall (i,j) \in \mathcal{P}, \tag{4}$$

$$S_i \geq r_i \qquad \forall i \in \mathcal{V}, \tag{5}$$

$$\sum_{i \in \mathcal{V}:\, S_i \leq t < S_i + d_i} q_{ik} \leq R_k \qquad \forall k \in \mathcal{R},\ \forall t. \tag{6}$$

In the Choco implementation the last family is posted through one `cumulative` propagator per renewable resource. Resource capacities and original activity demands are taken directly from PSPLIB; new activities generated by the disruption process are

inserted between the dummy source and sink as described in Section 3.5. For original PSPLIB activities, $r_i = 0$; for newly inserted activities, r_i is the release date of the snapshot at which the activity appears. The objectives are

$$C_{\max} = C_{n+1}, \qquad F_{\mathrm{RCPSP}} = \sum_{i \in \mathcal{V} \setminus \{0, n+1\}} (C_i - r_i). \tag{7}$$

Thus, when all releases are zero, the second objective is the total activity completion time; we keep the term *flowtime* only because the reactive-arrival setting creates nonzero releases for inserted activities.

3.4 Objectives and Schedule-Flexibility Diagnostic

We optimise two conflicting objectives, instrumented identically by every solver: the **makespan** $f_1 = C_{max} = \max_i C_{i,|J_i|}$ and the **total flowtime** $f_2 = F = \sum_i (C_{i,|J_i|} - r_i)$. The ε-constraint sweep is performed over (C_{max}, F), so the reported non-dominated sets and all hypervolume comparisons are genuinely two-objective. For RCPSP, the second objective is the released total activity completion time of Section 3.3, dummy source and sink excluded; the selectors, the shared ε-constraint sweep, and all hypervolume metrics are identical across the two families, only the resource propagator and the scheduled entities differ.

Schedule flexibility (reported, not optimised).

As a diagnostic we also record the average operational slack $\Phi = \frac{1}{|\mathcal{O}|} \sum_{O_{i,j}} (L_{start}(O_{i,j}) - E_{start}(O_{i,j}))$, with L_{start} relative to a relaxed bound $UB = 1.1 \times C^{opt}_{max}$. Φ measures *a priori* schedule flexibility, not realized robustness to disruptions; a genuine robustness objective would be $-\mathrm{Var}(C_{max})$ over K Monte-Carlo perturbation scenarios (Herroelen and Leus 2005), which multiplies the budget by K and is left to future work. Realized behaviour under disruption is captured instead by the schedule-stability analysis of Section 6.

3.5 Disruption Model

The same instance is unfolded into five snapshots $s_0, \ldots, s_4$ (indices $t = 0, \ldots, 4$). Snapshot s_0 is the nominal instance; each subsequent snapshot s_t is obtained from s_{t-1} by drawing, independently, from the two perturbation families below that modify the optimised problem. All draws use a generator seeded by (instance, snapshot, seed), so every heuristic sees byte-identical snapshots on a given (instance, seed), the basis of the paired design.

1. **Job arrivals.** The number of new jobs entering at snapshot t is $N_t \sim \mathrm{Poisson}(\lambda)$ with $\lambda = 0.5$. A new job-shop job receives one operation on every machine in a random order, with processing times drawn i.i.d. Uniform$\{1, \ldots, 15\}$; a new RCPSP activity receives a duration Uniform$\{1, \ldots, 10\}$ and, on each resource r, a demand Uniform$\{0, \ldots, \min(2, R_r)\}$ capped by the resource capacity R_r, and is inserted between the dummy source and sink (source→new→sink); precedence acyclicity

and resource feasibility are therefore preserved by construction (the PSPLIB j30/j60 capacities all satisfy $R_r \geq 2$, so the cap is inactive here).

2. **Machine breakdowns** (job-shop). A breakdown occurs at snapshot t with probability $p_{\text{bd}} = 1 - e^{-1/\text{MTBF}}$, where MTBF denotes the Mean Time Between Failures; MTBF $= 5$ (so $p_{\text{bd}} \approx 0.181$); it inflates the processing time of every operation on one randomly chosen machine by a factor Uniform$(1.5, 3.0)$, which is the time-extension form of a Mean Time To Recovery (MTTR) style outage on that resource. Because each snapshot derives from its predecessor, perturbations accumulate: an inflated machine stays inflated, and a second breakdown on the same machine compounds multiplicatively. For RCPSP the disruption set is limited to activity arrivals; resource-capacity shocks and duration perturbations are left to future work.

The generator also supports a job-priority perturbation (job weights perturbed by Uniform$(-0.3, 0.3)$ with probability 0.2 per snapshot). Because the two reported objectives, makespan and total flowtime, are *unweighted* (Section 3), this perturbation does not change the optimised problem or the Pareto front, so we do not count it as a disruption family; it would become an effective disruption only under a weighted flowtime objective $F_w = \sum_i w_i(C_i - r_i)$, which we leave to future work.

Snapshots are solved *independently*: operations are not frozen between snapshots but fully re-optimised, and the conflict weights are reset at each solve. This is the precise sense in which the study is "reactive re-solving" rather than industrial reactive scheduling: no operation is fixed as already in process, no schedule-repair constraint is posted, and no information is carried across disruptions (carrying weights across snapshots is future work, Section 7). The generator is likewise designed for controlled comparison rather than industrial realism, it does not model correlated failures, operator availability, or material shortages, so the conclusions are statements about search behaviour under controlled perturbations, not a validation of an industrial rescheduling system.

4 Conflict-Weighting Schemes and Experimental Protocol

4.1 Solver Pipeline

Each instance is unfolded as a sequence of T snapshots, each solved independently by an ε-constraint method (Haimes et al. 1971; Laborie et al. 2018) producing a non-dominated approximation in (C_{max}, F). The conflict-weighting selector under test acts as the variable selector inside Choco's branching tree; the value selector is the standard earliest-start (LB); the sweep produces at most κ points per snapshot, and the makespan-minimising point is retained for the flexibility and stability diagnostics; Algorithm 1 specifies the snapshot-level solve.

4.2 Conflict-Weighting Schemes and Their Exact Relationship

Let x be a CP decision variable, $\mathcal{C}(x)$ the set of propagators whose scope contains x, and $|D(x)|$ its current domain size. All conflict-directed selectors considered here pick

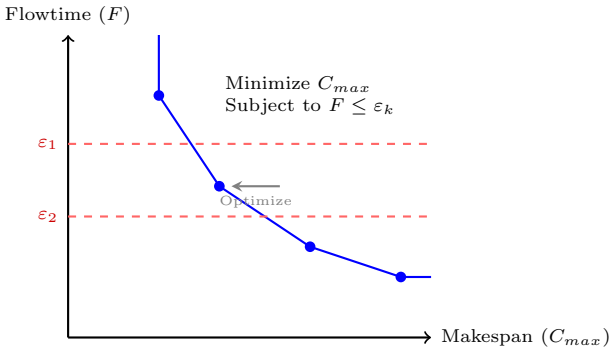


Fig. 1 The ε-constraint sweep: minimise C_{max} under a progressively tightened bound ε_k on F.

$\arg\max_x \text{score}(x)$ over unbound variables (ties broken by variable id) and differ only in how a contradiction caused by propagator c updates the weights and how those weights are summed into $\text{score}(x)$.

Scope-sharing (`dom/wdeg2004`).

Each constraint c carries a single scalar weight $w(c)$, incremented by one on every wipeout it causes. A variable is scored by

$$\text{score}_{2004}(x) = \frac{\sum_{c \in \mathcal{C}(x),\, |\text{fut}(c)|>1} w(c)}{|D(x)|}, \tag{8}$$

so the *full* weight of a failing constraint is credited to every variable in its scope. This is exactly the original weighted degree of Boussemart et al. (2004), and it is what Choco's `domOverWDegSearch` computes. The "scope-based" attribution is therefore not a generalisation of `dom/wdeg`, it *is* `dom/wdeg2004`.

AbsCon-inspired per-variable refinement.

Wattez et al. (2019) distribute conflict credit at a finer grain. The variant implemented here isolates the attribution component: on a wipeout of c, only the future (unbound) variables of c have their per-variable weight $w(c, x)$ incremented, and

$$\text{score}_{\text{pv}}(x) = \frac{\sum_{c \in \mathcal{C}(x)} w(c, x)}{|D(x)|}, \tag{9}$$

the summation following the same active-constraint convention as Eq. (8); in the implementation the increment rule is the only difference between the two selectors. We do not claim to reproduce every component of the full AbsCon heuristic, which also includes additional arity and domain-size considerations. We therefore label the method *AbsCon-inspired per-variable wdeg*. Explanation-based (Hébrard and Siala 2017) and conflict-history (Habet and Terrioux 2019) variants refine the credit and its temporal decay further. These are the genuine descendants of the 2004 rule.

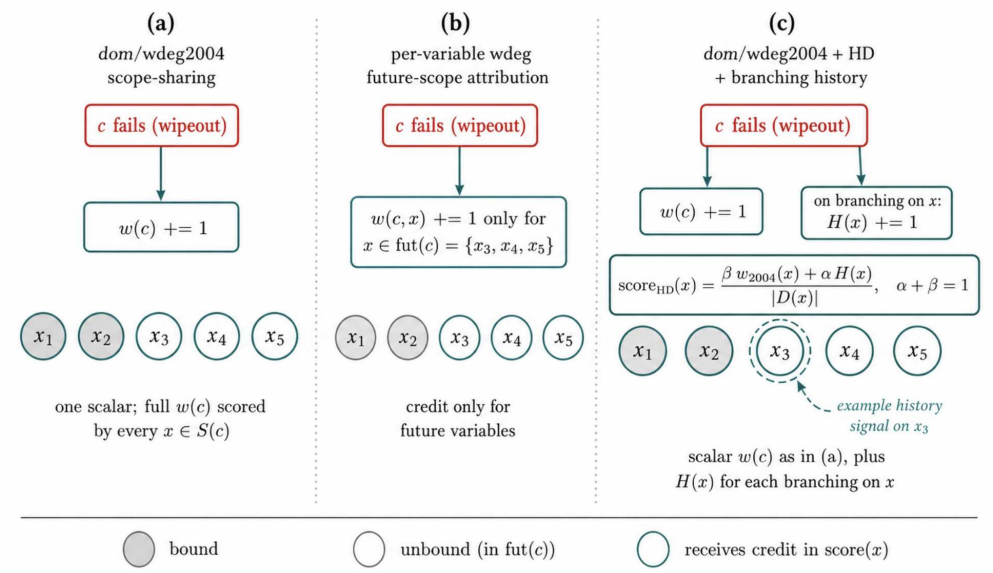


Fig. 2 Credit of one propagator failure under the three increment rules (Eqs. (8)–(10)); x_1, x_2 are bound, $\mathrm{fut}(c) = \{x_3, x_4, x_5\}$. (a) Scope-sharing increments one scalar $w(c)$, and the full weight enters the score of *every* scope variable (Eq. (8)). (b) The per-variable refinement credits only the future variables $\mathrm{fut}(c)$ (Eq. (9)). (c) HD keeps the scalar of (a) and adds a decision-driven term $\alpha H(x)$ for each branching on x (Eq. (10)). The lower the scope-freshness at failures, the more (a) and (b) diverge (Proposition 1).

Branching-history augmentation (HD).

The only scheme in our study that departs algorithmically from the above adds a branching-frequency term (HD, for *history of decisions*): with $H(x)$ the number of times x has been selected for branching,

$$\mathrm{score}_{\mathrm{HD}}(x) = \frac{\beta\, w_{2004}(x) + \alpha\, H(x)}{|D(x)|}, \qquad \alpha + \beta = 1. \tag{10}$$

Setting $\alpha = 0$ recovers (8) exactly. All reported experiments use $\alpha = \beta = 0.5$; Appendix B shows the conclusions are insensitive to $\alpha \in \{0.1, \ldots, 0.7\}$, which is reassuring since $w_{2004}(x)$ and $H(x)$ are combined without separate normalisation, and consistent with HD being a second-order term. HD is the only term retained from an earlier, more elaborate prototype (Appendix A) and is the single genuine algorithmic variation to be tested.

The rationale for HD is the same intuition that underlies activity-based search (Michel and Van Hentenryck 2012) and the VSIDS family of SAT heuristics: where `dom/wdeg` credits a variable only when a constraint in its scope *fails*, $H(x)$ credits it whenever the search *chooses* it, adding a self-reinforcing, decision-driven signal on top of the failure-driven one. This is a heuristic conjecture, not a guarantee (the same self-reinforcement can also lock the search into an unproductive region), which is why we treat HD as a hypothesis to be tested on branching-sensitive statistics (Section 6.4)

rather than asserted. Figure 2 contrasts the three increment rules on a single failing propagator.

Table 2 The three increment rules; everything else in the harness is identical. Rows 1 and 3 coincide when $\alpha = 0$.

Scheme	**On wipeout of** c	$\text{score}(x) \cdot \lvert D(x)\rvert$
`dom/wdeg2004` (scope-sharing)	$w(c)$ += 1 (scalar)	$\sum_{c \ni x,\ \lvert\text{fut}(c)\rvert>1} w(c)$
AbsCon-inspired per-variable	$w(c,x)$ += 1, $x \in \text{fut}(c)$	$\sum_{c \ni x} w(c,x)$
HD-augmented (this study)	$w(c)$ += 1 (scalar)	$\beta \sum_{c \ni x} w(c) + \alpha H(x)$

Implementation. All three schemes are implemented in a single Choco `VariableSelector` parametrised by the increment rule, plugged to an `IMonitorContradiction` listener, so that switching scheme changes nothing else in the search. This is what makes the comparison of Section 6 controlled.

Algorithm 1: Snapshot-level shared-grid ε-constraint multi-objective solve

Input: Snapshot instance s_t, conflict-weighting scheme, cached shared grid $\mathcal{E}_t = \{\varepsilon_0, \ldots, \varepsilon_{\kappa-1}\}$
Output: Non-dominated ε-constraint approximation $\mathcal{P}_t$ in (C_{max}, F), representative starts

$\mathcal{P}_t \leftarrow \emptyset$;
for $\varepsilon_k \in \mathcal{E}_t$ **do**
  reset all conflict weights to their initial value $w_0 = 1$ and reset branching histories;
  post $F \le \varepsilon_k$; minimise C_{max}, branching with the selected scheme; value selector LB;
  if *a feasible schedule with value* $C^\star_{max}$ *is returned* **then**
    optionally polish lexicographically by minimising F subject to $F \le \varepsilon_k$ and $C_{max} = C^\star_{max}$ within the remaining subproblem budget;
    add the best returned schedule to $\mathcal{P}_t$; `// infeasible` ε_k `skipped`
Remove dominated and duplicate points from $\mathcal{P}_t$;
return $\mathcal{P}_t$ *and the makespan-minimizing representative*;

The ε-grid is shared by all CP selectors for a given (instance, snapshot, seed): it is computed once from cached reference bound solves and reused by every selector, so the controlled comparison does not mix increment-rule effects with different ε thresholds. Conflict weights and HD branching histories are reset at the start of every ε subproblem and at every snapshot; no weight information crosses bound solves, ε points, or snapshots. Infeasible ε-subproblems contribute no point; a 2D non-dominated filter is applied before hypervolume is computed, and a snapshot with no feasible return is assigned HV = 0 (the same convention applies to a metaheuristic returning no feasible individual).

4.3 A Convergence Criterion for Scope-Sharing and Per-Variable Weighting

Proposition 1 characterises the condition under which scope-sharing and per-variable weighting would converge; it does not assume the condition holds for scheduling propagators, but makes it measurable. Fix a constraint c with scope $S(c)$, having caused $k(c)$ wipeouts so far, indexed $f = 1, \ldots, k(c)$, and let $\mathrm{fut}_f(c) \subseteq S(c)$ be the set of variables of c still unbound at wipeout f. The two schemes accumulate, for a variable $x \in S(c)$,

$$\underbrace{k(c)}_{\text{scope-sharing (2004)}} = \sum_{f=1}^{k(c)} 1, \qquad \underbrace{a(c,x)}_{\text{per-variable}} = \sum_{f=1}^{k(c)} \mathbf{1}[x \in \mathrm{fut}_f(c)], \tag{11}$$

so that $0 \le k(c) - a(c,x)$ equals the number of wipeouts of c at which x was *already bound*. Define the constraint's *scope-freshness* $b(c) = \min_{x \in S(c)} a(c,x)/k(c) \in [0,1]$, the worst-case fraction of c's failures at which a scope variable was still free, and let $\varepsilon = 1 - b(c)$.

Proposition 1 *Consider the scores restricted to the constraints currently active at a node (those with $|\mathrm{fut}(c)| > 1$), and suppose every such constraint satisfies $1 - b(c) \le \varepsilon$. Then for every unbound variable x,*

$$\left|\, \mathrm{score}_{\mathrm{pv}}(x) - \mathrm{score}_{2004}(x) \,\right| \;\le\; \varepsilon \cdot \mathrm{score}_{2004}(x). \tag{12}$$

Equivalently, $(1-\varepsilon)\,\mathrm{score}_{2004}(x) \le \mathrm{score}_{\mathrm{pv}}(x) \le \mathrm{score}_{2004}(x)$. Consequently the score_{2004}-best variable $x^\star$ remains the $\mathrm{score}_{\mathrm{pv}}$-best variable whenever

$$\frac{\mathrm{score}_{2004}(x^\star)}{\mathrm{score}_{2004}(y)} \;>\; \frac{1}{1-\varepsilon} \qquad \textit{for every competitor } y. \tag{13}$$

Proof For each active constraint $c \ni x$ we have $0 \le k(c) - a(c,x) \le (1 - b(c))\,k(c) \le \varepsilon\, k(c)$. Summing over the constraints containing x and dividing by $|D(x)|$,

$$\begin{aligned} 0 \;\le\; \mathrm{score}_{2004}(x) - \mathrm{score}_{\mathrm{pv}}(x) &= \frac{\sum_{c \ni x} \big(k(c) - a(c,x)\big)}{|D(x)|} \\ &\le \frac{\varepsilon \sum_{c \ni x} k(c)}{|D(x)|} = \varepsilon\, \mathrm{score}_{2004}(x), \end{aligned}$$

which is the stated bound and gives $(1-\varepsilon)\,\mathrm{score}_{2004}(x) \le \mathrm{score}_{\mathrm{pv}}(x) \le \mathrm{score}_{2004}(x)$. For the ranking claim, suppose (13) holds. Then for every y, $\mathrm{score}_{\mathrm{pv}}(x^\star) \ge (1-\varepsilon)\,\mathrm{score}_{2004}(x^\star) > \mathrm{score}_{2004}(y) \ge \mathrm{score}_{\mathrm{pv}}(y)$, so $x^\star$ is also pv-best. □

Premise, measurement, and their link.

For clarity the statement initialises all weights at zero, so that $w(c) = k(c)$; the implementation follows the usual convention $w(c) = w_0 + k(c)$ with $w_0 = 1$, and a common additive offset on both schemes does not change the difference $k(c) - a(c,x)$ that drives the bound. The proposition requires the freshness condition $1 - b(c) \le \varepsilon$ to

hold *uniformly* over the constraints active at a node, and its guarantee is meaningful only for *small* ε (the threshold $1/(1-\varepsilon)$ in (13) blows up as $\varepsilon \to 1$). The quantity we instrument and report (Table 8) is the *mean instantaneous* scope-freshness $\bar{\phi}$ over failures, where $\phi_f(c) = |\mathrm{fut}_f(c)|/|S(c)|$ is the fraction of c's scope still unbound at failure f, a different quantity from $b(c)$, but linked to it: summing the indicator $\mathbf{1}[x \in \mathrm{fut}_f(c)]$ over the scope gives $\sum_{x \in S(c)} a(c,x) = \sum_{f=1}^{k(c)} |\mathrm{fut}_f(c)|$, so the scope-average of $a(c,x)/k(c)$ equals the mean instantaneous freshness $\bar{\phi}(c)$ of c's failures, and in particular

$$b(c) \;=\; \min_{x \in S(c)} \frac{a(c,x)}{k(c)} \;\le\; \bar{\phi}(c). \tag{14}$$

Contrary to the intuition that long-scope `noOverlap`/`cumulative` propagators fail while most of their scope is free, the measured freshness is low on both families ($\bar{\phi} \approx 0.17$ job-shop, ≈ 0.30 RCPSP; Section 6.4): contradictions are detected late, when most start variables are already bound. By (14) the uniform premise could then hold only with $\varepsilon \ge 1 - \bar{\phi} \approx 0.7$–$0.83$, where the ranking threshold $1/(1-\varepsilon)$ is 3.3–5.9, the sufficient high-freshness convergence bound is vacuous on these benchmarks. High freshness being only *sufficient* for closeness, this does not by itself prove the rules differ (they could still coincide through score gaps or tie-breaking); it is the directly measured search-tree divergence of Section 6.4 that shows they are not interchangeable in the tested regime. We therefore read Proposition 1 as a falsifiable convergence *criterion*, not a two-sided guarantee; the equivalence we do observe in the main campaign (Section 6.1) is the one between pure `dom/wdeg2004` and Choco's `domOverWDeg`, which implement the same increment rule.

5 Experimental Setup

5.1 Solver and Hardware

The branching strategies are implemented in Choco 4.10.14 (Prud'homme and Fages 2022) (Java, JDK 11+), with each variable selector exposed as a `VariableSelector` that registers a contradiction listener (`IMonitorContradiction`) to maintain the conflict weights (Section 4.2). The CP model uses `noOverlap` on each machine for the job-shop, `cumulative` on each renewable resource for RCPSP, and a true makespan defined as the maximum completion time. A Python orchestrator unfolds each instance into snapshots, calls the Java fat-jar over each snapshot through a sub-process bridge, and aggregates the metrics. All controlled conflict-weighting selectors share the same model, the same default Choco propagation settings, the same earliest-start (LB) value selector, and run *without restarts* and with only the per-subproblem time limit as a search limit; the sole difference between these selectors is the weight-increment rule. For the activity baseline we use Choco's activity score update and decay, but disable restart schedules so that the baseline is compared under the same no-restart solve protocol. This choice has a known cost: both `dom/wdeg` and `activity` are commonly deployed *with* restarts (Grimes and Hébrard 2015; Michel and Van Hentenryck 2012), and the no-restart protocol is expected to penalise `activity` most, since it was designed around restart-driven score stabilisation. It is nevertheless what

makes the increment rule the only varying factor; the large effects reported against `activity` (Section 6.1) are therefore configuration-specific, not a verdict on activity-with-restarts. Initial weights are $w_0 = 1$, ties are broken by variable identifier, objective constraints are excluded from the failure-weight accounting, and null/non-propagator contradiction causes are ignored rather than attributed to a scheduling propagator. Weights attach to propagators (the `noOverlap` and `cumulative` objects reported as contradiction causes), which for these global constraints coincide one-to-one with the model constraints, so "constraint" and "propagator" may be read interchangeably here. Hardware details are given below.

5.2 Benchmarks

We use two families of standard instances:

- **Job-Shop:** 82 classical JSSP instances from OR-Library, Lawrence `la01-la40` (40), Fisher-Thompson `ft06/ft10/ft20` (3), Adams-Balas-Zawack `abz5-abz9` (5), ORB `orb01-orb10` (10), Storer-Wu-Vaccari `swv01-swv20` (20), and Yamada-Nakano `yn1-yn4` (4). The triple `la06--la08` is reserved for tuning the HD weight α and is *excluded* from the test set.
- **RCPSP:** PSPLIB (Kolisch and Sprecher 1997) `j30` and `j60` families: the ten instances of each of four consecutive parameter groups per family (`j30` 10–13 and `j60` 10–13), i.e. 40 instances per family (80 in total), spanning several network-complexity and resource-strength settings. The selection was fixed before any experiment and is part of the released instance lists; the full 480-instance sets were beyond the paired-design budget (6 methods × 25 seeded snapshots × 30 s per instance). None are used for tuning.

The dataset accounting is summarised in Table 3: $82 + 80 = 162$ standard instances, of which $79 + 80 = 159$ form the held-out test set on which all reported statistics are computed.

Table 3 Dataset accounting. HV/efficiency statistics use the test set; the diagnostic of Section 6.4 uses all 80 RCPSP instances and a stratified 23-instance job-shop subset.

Family	**Standard instances**	**Tuning excluded**	**Test n**
JSSP (la/ft/abz/orb/swv/yn)	82	3	79
RCPSP `j30`	40	0	40
RCPSP `j60`	40	0	40
Total	162	3	159

5.3 Reactive Disruptions

Each instance is unfolded into $T = 5$ reactive snapshots using the disruption model of Section 3.5 (Poisson job arrivals with $\lambda = 0.5$; exponential-MTBF machine breakdowns, MTBF = 5, realised as processing-time inflation). Perturbations are seeded by

(instance, snapshot, seed) so that all heuristics see byte-identical snapshots on a given (instance, seed), ensuring a paired-comparison design. Snapshots are re-optimised independently with conflict weights reset at each solve, so "reactive" here denotes repeated re-solving of perturbed snapshots, not learning across them.

5.4 Baselines and Heuristics

Table 4 fixes the label used for each method throughout the paper. These labels are used consistently in all tables, figures, and captions; the internal implementation name used in earlier prototypes is not used as a method label in the final analysis.

Table 4 Method labels used throughout the paper.

Label	Implementation / meaning
`dom`	Choco `minDomLBSearch`
`activity`	Choco activity-based search
Choco `dom/wdeg`	Choco `domOverWDegSearch`; implements `dom/wdeg2004`
`dom/wdeg2004`	Original scope-sharing weighted-degree rule
AbsCon-inspired per-variable wdeg	Per-variable conflict-weighting refinement
dom/wdeg2004+HD	`dom/wdeg2004` plus branching-history term
NSGA-II	Population-based multi-objective metaheuristic
NSGA-III	Reference-direction multi-objective metaheuristic

The study runs at two levels. The *main campaign* compares three native CP variable selectors, `dom`, Choco `dom/wdeg`, and `activity` (Michel and Van Hentenryck 2012), the HD-augmented variant evaluated in this study (*dom/wdeg2004+HD*, Eq. (10)), and the NSGA-II/NSGA-III metaheuristics (Deb et al. 2002; Deb and Jain 2014).

Metaheuristic baselines.

The NSGA implementations (pymoo) use a *random-key* encoding: a chromosome is a real vector of length n (one key per job/activity), decoded by `argsort` into a job-priority order, which a serial schedule-generation scheme turns into a feasible schedule; no repair or penalty is needed. This is a *priority-rule* encoding: for the job-shop it searches over job-priority orderings rather than the full space of operation sequences, so NSGA-II/III are priority-rule baselines, not general-purpose job-shop solvers, appropriate as a contextual reference for the CP selectors, and flagged as a limitation of the metaheuristic comparison. The headline NSGA runs optimise exactly the two objectives used for hypervolume, $(C_{\max}, F)$; a three-objective variant including $-\Phi$ is retained only as a supplementary diagnostic. NSGA-II uses a population of 50; NSGA-III uses Das-Dennis reference directions with $n_{\text{partitions}} = 12$ (population 91); both use simulated-binary crossover and polynomial mutation (pymoo defaults), time-based termination equal to the per-snapshot budget, and the same paired seeds. A broader solver comparison (CP-SAT, CP Optimizer, MOEA/D, reactive dispatching rules) is left to future work.

A *controlled weighting experiment* (Section 6.4) then isolates the increment rule: the three schemes of Table 2 (`dom/wdeg2004`, per-variable wdeg, HD-augmented) are run in a single Choco harness in which everything but the increment rule is held fixed.

Fairness of the CP-NSGA comparison.

The CP selectors and the metaheuristics build their fronts by different mechanisms, and we make the comparison fair on the axes that matter. (i) *Same total per-snapshot budget*: every method receives the identical 30 s wall-clock budget per snapshot, single-threaded, on the same hardware; for the CP solver this total is divided across the internal solves of the ε-sweep (each subproblem receives $30/(\kappa+2) \approx 4.3$ s), while NSGA-II/III consume their 30 s as a single population run. (ii) *Same objectives*: hypervolume is computed for every method on the two shared objectives $(C_{\max}, F)$, instrumented identically and optimised by all headline runs. (iii) *Same normalised reference*: a per-(instance, snapshot) ideal/nadir and reference point shared across all methods (Section 5.6), so no method can shift its own baseline. (iv) *Front cardinality is not scored*: we rank on dominated hypervolume only, and Section 6.1 additionally verifies that capping the NSGA fronts to the CP front cardinality leaves the comparison unchanged. (v) The metaheuristic population sizes are the library defaults and were *not* reduced in CP's favour. "CP vs NSGA" is read throughout as a quality-at-equal-budget trade-off within this priority-rule comparison, not as a claim that one front-construction mechanism dominates the other in the abstract.

5.5 Protocol and Reproducibility

Each heuristic is run with 5 seeds in $[0, 4]$ on every instance and snapshot, all seeds identical across heuristics within an instance; over the test set (Table 3) this gives $79 \times 5 \times 5 = 1975$ job-shop and $80 \times 5 \times 5 = 2000$ RCPSP paired blocks, aggregated to one value per (heuristic, instance) before testing. The time budget is 30 s *total per snapshot* for every method. Budget accounting: the ε-grid construction (the two bound solves) is performed once per (instance, snapshot, seed) and charged within the snapshot budget; the cached grid is then reused by every CP selector, whose budget covers the $\kappa = 5$ ε-points, each with a hard limit set through Choco's `limitTime` (an outer subprocess timeout guards against hangs); NSGA-II/III consume the same 30 s as a single run. Table 7 reports Choco's internal solver time (`getTimeCount`) summed over a selector's solves; Python orchestration and JVM start-up are excluded from that figure but captured by the end-to-end wall-clock timers released with the logs, and NSGA times are end-to-end. The HD weight α and the now-removed auxiliary terms were tuned by random search (50 iterations) on the three tuning instances (`la06`–`la08`), excluded from every test set. Disruptions come from a generator seeded by (instance, snapshot, seed), so all heuristics see byte-identical snapshots, the basis of the paired design. The benchmarks are public (OR-Library, PSPLIB); solver: Choco 4.10.14. The full source, instance lists, seeds, canonical result CSV files, and analysis scripts are released to allow exact reproduction; all numerical tables are generated from the same canonical results directory, produced under this shared-grid protocol.

Controlled-experiment budget.

The controlled weighting experiment of Section 6.4 is a search-behaviour diagnostic, not a front-quality run; it uses its own fixed budget, 3 s per ε-point and 2 ε-points in the single instrumented harness, so that the three increment rules are compared at an identical, clearly stated budget on every solve. It is reported on the full RCPSP set and on a stratified job-shop subset spanning all six families (`la`, `ft`, `abz`, `orb`, `swv`, `yn`); the released `ExperimentWeighting` harness reproduces it on any instance.

Hardware and software.

All experiments were run single-threaded on a MacBook Pro with an Apple M1 Pro CPU (8 performance cores), 16 GB RAM, under macOS 14, using the Eclipse Temurin OpenJDK 11 runtime; Python 3.11 drove the orchestrator and analysis.

Reproducibility.

The repository accompanying this paper includes the Choco implementation of all selectors, the Python orchestration scripts, the disruption generator, the benchmark instance lists, the random seeds, the raw result files, and the plotting and statistical-analysis scripts. An anonymised reproducibility package (code, instance lists, seeds, raw results, and analysis scripts) is provided as supplementary material at submission, and the repository will be made public upon acceptance.

5.6 Metrics

The primary metric is the **hypervolume (HV)** (Zitzler and Thiele 1999) of the returned non-dominated set. With the headline $\kappa = 5$ shared grid the CP solver produces a *sampled ε-constraint approximation* (at most κ points), each ε-subproblem minimising $C_{\max}$ under $F \leq \varepsilon$ with an optional lexicographic polishing step on F when the budget permits (without polishing, points are efficient only up to weak efficiency); NSGA-II/III return a population-sized approximation. We therefore use "non-dominated ε-constraint approximation" for the CP output, reserve "Pareto front" for the exact notion, and compare methods only through dominated hypervolume, computed on the two identically instrumented objectives, makespan and flowtime, normalised per (instance, snapshot). Let P_h be the front returned by heuristic h on a given (instance, snapshot), and let $U = \bigcup_h P_h$ be the union of all fronts over the heuristics and seeds of that (instance, snapshot). The shared ideal and nadir points are

$$z_j^{\text{ideal}} = \min_{p \in U} p_j, \qquad z_j^{\text{nadir}} = \max_{p \in U} p_j, \qquad j \in \{\text{makespan}, \text{flowtime}\}, \tag{15}$$

and each point is mapped to $\tilde{p}_j = (p_j - z_j^{\text{ideal}}) / \max(z_j^{\text{nadir}} - z_j^{\text{ideal}}, 1)$. The normalised HV is the area of $[0, r]^2$ dominated by the normalised front, with reference point

$r = 1.1$ on each axis:

$$\mathrm{HV}(P_h) = \lambda\left(\bigcup_{\tilde{p}\in\tilde{P}_h} [\tilde{p}_1, r] \times [\tilde{p}_2, r]\right), \tag{16}$$

where λ is the Lebesgue measure. Because $z^{\mathrm{ideal}}, z^{\mathrm{nadir}}$ and r are *shared* across all heuristics on that (instance, snapshot), the comparison is paired and scale-free; a heuristic absent from U cannot shift the reference, and conclusions are insensitive to $r \in \{1.05, 1.1, 1.2\}$ and to using the pooled non-dominated union for ideal/nadir construction. The metrics analysed statistically are **HV**, **CPU time**, and **search-tree statistics** (nodes, failures, closure rate; Section 6.4); **schedule stability** is reported descriptively (Section 6). Front *spacing*, front *size*, and the flexibility diagnostic Φ are recorded but never used to rank solvers (they reflect the front-construction mechanism rather than quality) and are provided in the supplementary material.

5.7 Statistical Tests

To avoid pseudo-replication, the statistical unit is the *instance*: snapshots are repeated measures and seeds are paired, so each metric is aggregated to one value per (heuristic, instance) before testing, giving $n = 79$ job-shop and $n = 80$ RCPSP paired observations. A Friedman test ranks the heuristics; a Nemenyi post-hoc with critical-distance diagram visualises pairwise differences; paired Wilcoxon signed-rank tests compare the reference selector to each baseline, with p-values Holm-corrected across the five comparisons within each family. Alongside each p-value we report the Vargha-Delaney $\hat{A}_{12}$ effect size between the two per-instance samples (ties counted half); the paired structure is captured by the Wilcoxon tests and the bootstrap confidence intervals on median paired differences (Table 5), so significance and practical magnitude are never conflated. Families are reported separately.

6 Results and Discussion

6.1 Hypervolume Comparison and Statistical Significance

The Friedman test rejects equivalence on both families (job-shop $\chi^2 = 343.9$, $p \approx 4\times10^{-72}$; RCPSP $\chi^2 = 108.1$, $p \approx 10^{-21}$; instance as unit, Section 5). Figures 4 and 5 report the Nemenyi critical-distance diagrams; Figure 3 shows the HV distributions. Table 5 reports the Holm-corrected paired Wilcoxon tests, Vargha-Delaney $\hat{A}_{12}$ effect sizes (> 0.5 favouring the HD-augmented selector, *dom/wdeg2004+HD*), and median paired differences with 95% bootstrap confidence intervals.

The picture is as follows. The HD-augmented selector and Choco `dom/wdeg` are close on hypervolume on both families, an exact tie on RCPSP, and a small but Holm-significant gap on job-shop ($\hat{A}_{12} = 0.47$, $p = 0.024$) where adding HD is marginally worse.

The exactness of the RCPSP tie deserves explanation, since HD does change the explored tree (Section 6.4). It is not an aggregation artefact: the two selectors

Table 5 Front quality vs *dom/wdeg2004+HD* (normalised HV, instance as unit): Holm-corrected Wilcoxon p, $\hat{A}_{12}$ (> 0.5 favours HD-augmented; thresholds on $\max(\hat{A}_{12}, 1 - \hat{A}_{12})$), median paired Δ [95% bootstrap CI].

Family	**Baseline**	**Holm p**	**$\hat{A}_{12}$ (effect)**	**median Δ [95% CI]**
Job-shop (n=79)	`dom`	4.5×10^{-12}	0.549 (negligible)	+0.026 [+0.017, +0.035]
	Choco `dom/wdeg`	2.4×10^{-2}	0.470 (negligible)	−0.015 [−0.027, +0.001]
	`activity`	3.2×10^{-13}	0.776 (large)	+0.223 [+0.127, +0.293]
	NSGA-II	5.7×10^{-14}	0.035 (large, NSGA)	−0.537 [−0.838, −0.332]
	NSGA-III	4.6×10^{-14}	0.036 (large, NSGA)	−0.534 [−0.846, −0.343]
RCPSP (n=80)	`dom`	4.8×10^{-2}	0.457 (negligible)	0.000 [0.000, 0.000]
	Choco `dom/wdeg`	1.00	0.500 (exact tie)	0.000 [0.000, 0.000]
	`activity`	8.2×10^{-10}	0.826 (large)	+1.002 [+0.941, +1.043]
	NSGA-II	5.4×10^{-8}	0.692 (medium)	+0.499 [+0.010, +0.977]
	NSGA-III	4.1×10^{-8}	0.692 (medium)	+0.499 [+0.044, +0.977]

return identical normalised HV on every one of the 2000 (instance, snapshot, seed) RCPSP blocks, and their deduplicated termination counts on the $2 \times 2000 = 4000$ mono-objective bound solves are identical as well (2900 proved optimal, 640 feasible at the limit, 460 without a solution, a 72.5% proof rate). The mechanism is the campaign regime: under the 30 s budget the RCPSP ε-subproblems overwhelmingly close (median solver time 0.24–0.27 s, Table 7), a closed subproblem returns the same optimal objective vector for any complete selector, and on the minority of solves stopping at the limit the returned incumbents produced identical hypervolume under the shared normalisation. The tie is thus a statement about the non-binding RCPSP budget, HD reorders the tree without changing what is returned, not a claim that the two searches are identical. Under the no-restart protocol used to isolate the weighting rule, both clearly beat `activity` with a large effect ($\hat{A}_{12} = 0.78$ job-shop, 0.83 RCPSP; Section 5 discusses why this configuration penalises `activity` most). Relative to `dom` the comparison is *family-dependent*: on job-shop the conflict-weighting selectors are better with a small effect (Choco `dom/wdeg` vs `dom` gives $\hat{A}_{12} = 0.58$, Holm $p \approx 2 \times 10^{-9}$), whereas on RCPSP `dom` is slightly *higher* ($\hat{A}_{12} = 0.457$, Holm $p = 0.048$) and the difference is negligible in magnitude. Against the metaheuristics the story is also family-dependent: on job-shop NSGA-II/III reach a higher normalised HV (large effect, $\hat{A}_{12} \approx 0.04$ against them), while on RCPSP the CP selectors *beat* both NSGA variants with a medium effect ($\hat{A}_{12} = 0.69$). The job-shop gap is not a front-cardinality artefact: reducing each NSGA front to its best five points (greedy hypervolume subset, same shared normalisation) changes its per-instance HV by at most 0.002 (median 10^{-4}) and leaves the capped NSGA-II front above the HD-augmented selector on all 79 test instances. The interpretation is rather that, on the harder job-shop instances, few CP ε-subproblems close within the budget, so the population-based search covers more of the objective space; the advantage of CP appears primarily in proof capability, for the subproblems that close within the limit, and in RCPSP solver effort (Section 6.3).

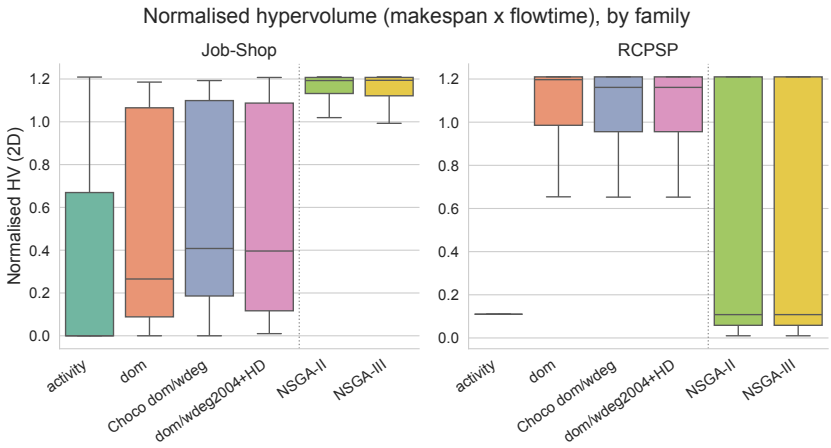


Fig. 3 Normalised HV by heuristic and family. Maximum attainable value is $r^2 = 1.21$.

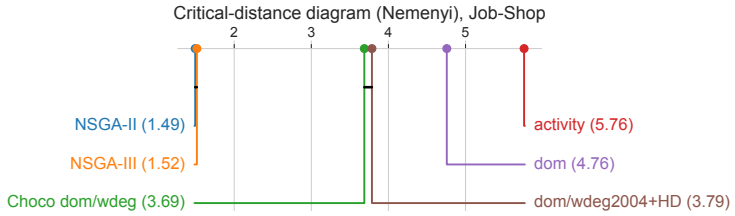


Fig. 4 Nemenyi critical-distance diagram on normalised HV, job-shop.

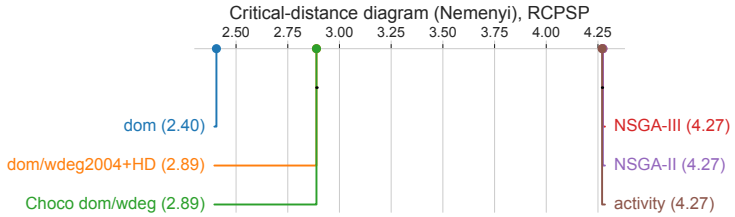


Fig. 5 Nemenyi critical-distance diagram on normalised HV, RCPSP.

6.2 Ablation of the HD Term on the Final Normalised Metric

HD is the only retained augmentation; additional discarded prototype terms are documented in Appendix A. We ablate HD directly, on the same family-comparable normalised hypervolume used throughout (Section 5.6), with the instance as the statistical unit. Table 6 compares `dom/wdeg2004` with and without HD, against `dom` and `activity`.

HD does not improve hypervolume. On RCPSP the +HD and no-HD configurations return *identical* per-instance HV (an exact tie, zero paired differences); on job-shop +HD is marginally *worse* ($\Delta = -0.023$, $p = 0.024$, negligible effect

Table 6 HD ablation on normalised HV (mean per family; instance as unit): change Δ, paired Wilcoxon p, $\hat{A}_{12}$ of +HD vs no-HD.

Family	`dom`	`dom/wdeg2004`	+HD	`activity`	Δ HD	p	$\hat{A}_{12}$
Job-shop	0.530	0.599	0.576	0.275	−0.023	0.024	0.47 (negligible)
RCPSP	1.005	0.928	0.928	0.272	0.000	,	0.50 (exact tie)

$\hat{A}_{12} = 0.47$). The "HD contributes most" reading of an earlier non-normalised three-coordinate proxy (Appendix A) does not survive on the final metric. Where HD does have a measurable effect is search effort, isolated in the controlled experiment of Section 6.4: under a fixed budget, hypervolume cannot separate two schemes that reach similar fronts by exploring different numbers of nodes, which is why a node-level diagnostic is needed.

6.3 Efficiency Trade-off

Table 7 reports the per-instance median CPU per snapshot and the Friedman mean rank on CPU (lower is better; instance as unit); the timing conventions are stated in the caption. The picture is family-dependent. On job-shop the budget is binding on hard instances: the CP selectors consume a median of 28 s against the metaheuristics' 30 s, so we claim no time advantage there, the relevant CP property is anytime feasibility plus optimality where subproblems close, while NSGA attains higher hypervolume (Section 6.1). On RCPSP the gap survives the measurement caveat: `dom/wdeg2004+HD` returns its non-dominated set in a median of 0.24 s of solver time per snapshot against 30 s; the one-to-two orders-of-magnitude advantage is therefore an RCPSP, solver-time result. One row requires care: `activity`'s lower job-shop median (10 s) reflects failure, not speed, in 52% of its job-shop blocks it finds no feasible schedule within the subproblem limits, the ε-sweep terminating early with an empty return (scored HV = 0), against under 1% for the conflict-weighting selectors. Figure 6 positions every heuristic in the (mean CPU, mean normalised HV) plane.

Table 7 Median CPU per snapshot (s) and Friedman rank (lower is better). CP rows: Choco internal solver time; NSGA rows: end-to-end wall-clock.

	Job-shop		**RCPSP**	
Heuristic	**median (s)**	**rank**	**median (s)**	**rank**
dom/wdeg2004+HD	28.0	2.81	**0.24**	**1.88**
`dom`	28.0	2.68	0.25	2.43
Choco `dom/wdeg`	28.0	3.02	0.27	2.20
`activity`	10.0	1.49	1.56	3.50
NSGA-II	30.0	5.05	30.0	5.00
NSGA-III	30.1	5.95	30.0	5.99

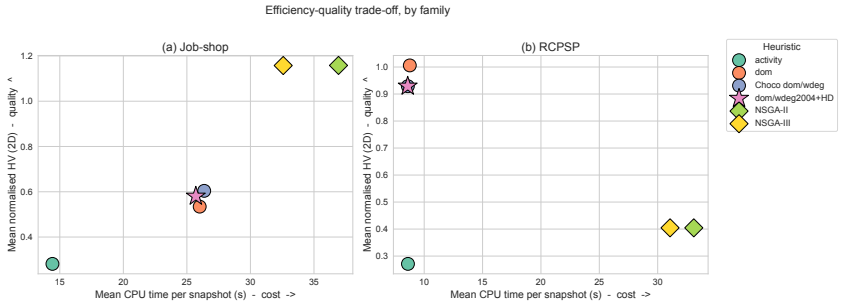


Fig. 6 Mean CPU vs mean normalised HV, by family.

6.4 Search Behaviour and Closure

Hypervolume under a fixed budget cannot, on its own, separate two weighting schemes that reach similar fronts by exploring different trees. We therefore run the controlled experiment of Table 2 in a single harness and measure branching-sensitive statistics, nodes and failures explored within the budget, closure rate, and the mean instantaneous scope-freshness $\bar{\phi}$ at failures (which by (14) upper-bounds the history freshness $b(c)$ of Proposition 1), reported alongside $1 - \bar{\phi}$. Table 8 gives the result on all 80 RCPSP instances and a stratified job-shop subset ($n = 23$, spanning all six families), at the fixed 3 s/ε-point diagnostic budget. On the ten `j6013` instances (the hardest j60 parameter group) *no* solve closes within the diagnostic budget for *any* variant, so node counts there measure budget-limited throughput rather than effort to a proof; we therefore report both the full set ($n = 80$) and the closing subset ($n = 70$), where node counts are interpretable as search effort.

Table 8 Controlled experiment (only the increment rule varies; 3 s/ε-point): closure, nodes, fails, and mean instantaneous scope-freshness $\bar{\phi}$ (Eq. (14) links it to Proposition 1). Closing subset excludes the ten `j6013` instances, on which no variant closes any solve.

Family	Selector	Closure %	Nodes	Fails	$\bar{\phi}$	$1 - \bar{\phi}$
Job-shop (n=23)	`dom/wdeg2004` (scope-sharing)	5.4	70,011	69,493	0.171	0.829
	AbsCon-inspired per-variable wdeg	5.4	79,799	79,263	0.203	0.797
	dom/wdeg2004+HD	4.3	59,107	58,603	0.188	0.812
RCPSP, full (n=80)	`dom/wdeg2004` (scope-sharing)	55.0	36,889	36,819	0.284	0.716
	AbsCon-inspired per-variable wdeg	56.6	39,870	39,812	0.312	0.688
	dom/wdeg2004+HD	57.5	37,867	37,797	0.296	0.704
RCPSP, closing (n=70)	`dom/wdeg2004` (scope-sharing)	62.9	30,463	30,413	0.300	0.700
	AbsCon-inspired per-variable wdeg	64.6	33,918	33,873	0.334	0.666
	dom/wdeg2004+HD	65.7	26,975	26,927	0.312	0.688

Three findings emerge. First, **the measured instantaneous scope-freshness is low**, $\bar{\phi} \approx 0.17$ (job-shop) and ≈ 0.28–0.30 (RCPSP), and the conclusion does not rest on the mean alone: Figure 7 shows the per-instance distributions, whose upper quartiles (0.25 job-shop, 0.38 RCPSP) and even maxima (0.37, 0.64) remain below the high-freshness regime. Contrary to the intuition that long-scope resource propagators

fail while most of their scope is free, on these benchmarks they fail late, when most start variables are already bound. Since $b(c) \leq \bar{\phi}(c)$ (Eq. (14)), the sufficient high-freshness convergence bound of Proposition 1 is vacuous here. This is necessary-side evidence only (the rules could still coincide through score gaps or tie-breaking), which is why the node-level divergence below is the decisive observation. Second, **scope-sharing and the AbsCon-inspired per-variable refinement are not automatically interchangeable**: the per-variable rule changes the weighting trajectory and produces different descriptive node/failure profiles (about 14% more nodes on job-shop, paired Wilcoxon $p = 0.052$, and 8–11% more on RCPSP at essentially the same closure), but the node-count differences are not statistically reliable on RCPSP ($p = 0.93$ full set, $p = 0.84$ closing subset) and are descriptive on job-shop, where closure is only ≈5% and node counts mostly measure budget-limited throughput. The equivalence reported on hypervolume in Section 6.1 is therefore strictly the one between pure `dom/wdeg2004` and Choco's `domOverWDeg`, which share the same increment rule. Third, **the HD term acts on search effort where proofs close, not on front quality**: it leaves hypervolume unchanged (Section 6.2), and on the closing RCPSP subset it reduces the nodes explored by about 11% (30,463 → 26,975, paired Wilcoxon $p = 0.019$) with a small closure gain (62.9% → 65.7%, $p = 0.059$). Figure 8 shows the per-instance picture honestly: about half the instances sit on or near the diagonal, and the mean reduction is driven by the closing `j60` instances. Over the full RCPSP set the closure gain persists (55.0% → 57.5%, $p = 0.059$) but the node reduction does not ($p = 0.72$), because the ten `j6013` instances do not close for any variant. Split by subfamily, closure is similar on `j30` and `j60` (55.6%/54.4% for scope-sharing), and the HD node reduction on the closing subset is concentrated on the closing `j60` instances (15,196 → 9,710 mean nodes, vs 41,913 → 39,925 on `j30`). On job-shop, where almost nothing closes, the approximate 16% node reduction is not statistically significant ($p = 0.11$). HD is thus a modest effect, confined to the RCPSP instances where proofs close.

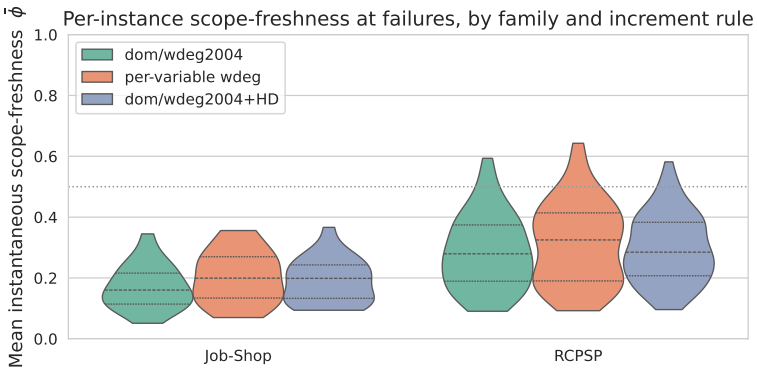


Fig. 7 Per-instance distribution of $\bar{\phi}$ (controlled harness); the whole distribution lies below the high-freshness regime of Proposition 1.

These results also reconcile the campaign-level termination profiles: on the 4000 RCPSP mono-objective bound solves, the HD-augmented selector and Choco

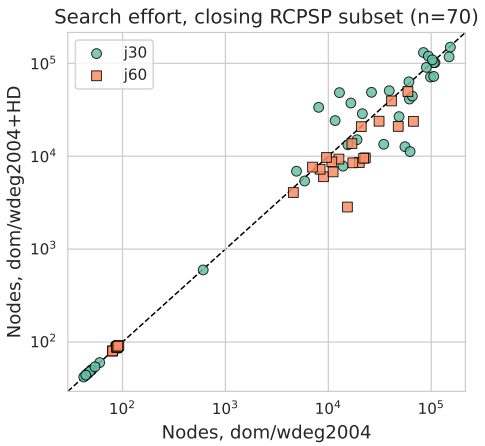


Fig. 8 Nodes, +HD vs `dom/wdeg2004`, closing RCPSP subset (n=70, log–log); points below the diagonal favour HD.

`dom/wdeg` log identical deduplicated termination counts (2900/640/460; Section 6.1). On job-shop the median CP solve sits at $\approx$ 28 s against the 30 s budget, so most job-shop solves stop at the limit without proof; CP's guarantee is therefore *anytime feasibility plus optimality where proved*, applying only to subproblems that close within the time budget.

6.5 Schedule Stability Analysis

Schedule stability between two consecutive snapshots is the average absolute deviation of start times over operations common to both schedules (newly arrived operations excluded), normalised by the nominal makespan; lower is more stable. Figure 9 shows the empirical cumulative distributions over (instance, seed) pairs for the four CP selectors on the job-shop family: the conflict-weighting selectors and `dom` are markedly more stable than `activity`, with strongly overlapping curves among themselves. Stability is reported for the job-shop CP selectors only (the campaign logs do not retain the cross-snapshot representative start vectors needed for RCPSP or for the NSGA baselines) and is a secondary, *descriptive* diagnostic from which we draw no statistical conclusion.

6.6 Sensitivity Analysis

We check that the conclusions do not hinge on the free parameters of the protocol; Table 9 summarises the four factors we varied and Appendix B gives the numbers. These checks are diagnostic, run on the full benchmark for the reference point, and on a small representative set for the harness-based factors, rather than a full benchmark rerun. The hypervolume reference point and the ε-grid size κ leave the selector ordering unchanged (with cost scaling predictably in κ); the time budget changes absolute closure but not the relative ranking; and the HD weight α affects node counts only within run-to-run noise, consistent with HD being a second-order, search-effort term.

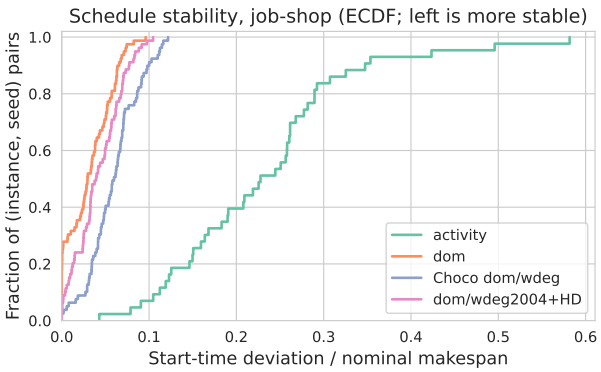


Fig. 9 Schedule stability, job-shop CP selectors: ECDF of start-time deviation between consecutive snapshots, normalised by nominal makespan (left/higher is more stable).

Disruption intensity was fixed at the nominal $\lambda = 0.5$ and is not varied here; sweeping it is noted as future work. The qualitative conclusions of Sections 6.1–6.4 hold throughout.

Table 9 Sensitivity summary (one factor varied at a time).

Factor	**Values tested**	**Main conclusion**
HV reference point r	1.05, 1.10, 1.20	selector ranking stable
ε-grid size κ	3, 5, 10 (subset)	ranking stable; cost scales with κ
Time budget (per ε-point)	3 s, 10 s	absolute closure changes, ranking stable
HD weight α	0.1, 0.3, 0.5, 0.7	affects nodes, not HV or closure

The sustained HV scores across snapshots are depicted in Figure 10.

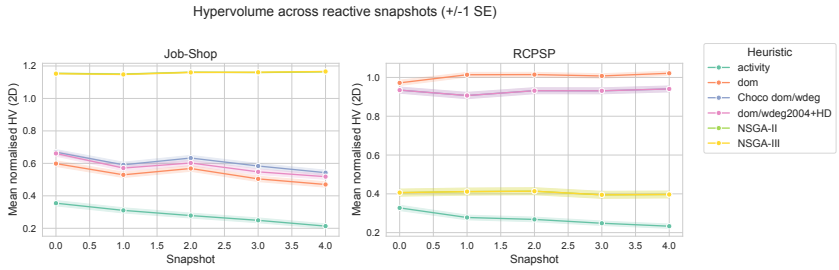


Fig. 10 Mean normalised HV per snapshot (±1 SE across instances).

7 Conclusion and Future Work

This paper provided a controlled study of conflict-weighting variable selection for repeated multi-objective job-shop and RCPSP re-solving. The first contribution is a clarification: crediting a failure to the entire scope of the failing propagator is the original `dom/wdeg2004` rule and corresponds to Choco's `domOverWDegSearch` (Section 4.2). The second is empirical: Choco `dom/wdeg` and pure `dom/wdeg2004` coincide up to tie-breaking; the HD-augmented variant is a distinct selector that is statistically a near-tie on hypervolume (exact tie on RCPSP, small but significant gap on job-shop); under the shared no-restart protocol both clearly outperform `activity`, while their advantage over `dom` is weak and family-dependent. The controlled search-tree experiment shows that the per-variable refinement changes the weighting trajectory and produces different descriptive node/failure profiles: measured instantaneous scope-freshness is low ($\bar{\phi} \approx 0.17$–0.30), making the sufficient high-freshness convergence bound of Proposition 1 vacuous here, although the node-count differences are not uniformly statistically significant. HD does not improve front quality; its one measurable effect is a node reduction confined to the RCPSP instances where proofs close (-11%, $p = 0.019$; closure $62.9\% \to 65.7\%$, $p = 0.059$). Compared with the contextual priority-rule NSGA-II/III baselines, CP selectors offer far lower internal solver time on RCPSP (0.24 s median against 30 s) and proofs for the subproblems that close, while NSGA achieves higher job-shop hypervolume even when capped to the CP front cardinality. Table 10 states explicitly what is and is not claimed.

Practitioner guidance. First, Choco's default `dom/wdeg` already implements the 2004 scope-sharing rule: a "scope-based" variant is not an upgrade but the same selector. Second, per-variable attribution changes the explored tree on long-scope propagators but should not be expected to change front quality under a budgeted ε-constraint sweep; it is a search-effort knob, not a quality knob. Third, decision-driven augmentations such as HD are worth considering only when proof closure rather than hypervolume is the binding concern (for instance, embedded re-solving where a proved-optimal schedule is required before dispatch). Family-wise, the budgeted ε-constraint CP approach is the clear choice on RCPSP-like models, whereas on hard job-shop instances, where few subproblems close, a population-based front under the same budget covers more of the objective space.

Limitations and threats to validity. (i) We optimise two objectives; the average-slack quantity Φ is an *a priori* flexibility diagnostic, not a realised-robustness objective, which would multiply the budget by the number of Monte-Carlo scenarios. (ii) "Reactive" here means controlled repeated re-optimisation: snapshots are solved independently, no operation is frozen as already in process, no machine-unavailability calendar is posted, and conflict weights are reset at each solve, so no information is carried across disruptions. (iii) Both models assume a single machine/resource assignment per operation; the flexible variants (Flexible Job-Shop, FJSP; Multi-Mode RCPSP, MRCPSP) are out of scope. (iv) The disruption model uses Poisson arrivals and an exponential-MTBF breakdown process realised as processing-time inflation, which does not capture correlated failures. (v) The controlled search-tree experiment runs on the full RCPSP set but on a stratified job-shop subset; the released harness reproduces it on the full set. (vi) We deliberately did not include CP-SAT or CP Optimizer: the

controlled design requires a single harness in which the weight-increment rule is the *only* varying factor, and a cross-solver comparison would confound modelling, propagation strength, and search engine with the increment rule under study. (vii) All CP selectors run *without restarts*; this isolates the increment rule but penalises `activity` most (it was designed around restart-driven score stabilisation), so that comparison is configuration-specific.

Future work. Natural extensions are: a Monte-Carlo variance-of-makespan robustness objective at a higher budget; extending the controlled comparison to the explanation-based (Hébrard and Siala 2017) and conflict-history (Habet and Terrioux 2019) weightings and to restart-enabled configurations; carrying conflict weights across snapshots with perturbation-aware decay instead of resetting them; and a previous-schedule-guided value selector using the representative schedule of snapshot $t-1$ as a value-ordering hint for snapshot t, a natural baseline for re-optimisation in practice.

Table 10 Claims and their support.

Claim	Supported by	Status
Scope-sharing is `dom/wdeg2004`	Algebra, Section 4.2	Established
Choco `dom/wdeg` ≡ pure `dom/wdeg2004` (up to tie-breaking)	HV (Sec. 6.1), RCPSP termination	Supported
AbsCon-inspired per-variable wdeg differs from scope-sharing; node-count differences are not uniformly significant	Proposition 1, Table 8	Diagnostic, cautious
HD improves hypervolume	Table 6	Not supported
HD reduces search effort on RCPSP where proofs close	Table 8 (n=70)	Supported, modest
HD reduces search effort on the full RCPSP set	Table 8 (n=80)	Not supported
CP universally dominates NSGA on HV	Table 5	Not supported
CP offers lower internal solver time and proof where closed	Table 7, Section 6.4	Supported
The method learns across snapshots	Protocol (weights reset)	Not claimed

Statements and Declarations

Funding. The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Competing interests. The authors have no relevant financial or non-financial interests to disclose.

Data and code availability. All benchmark instances are publicly available from OR-Library (job-shop families) and PSPLIB (j30/j60). The complete reproducibility package, the Choco implementation of all selectors, the Python orchestration and analysis scripts, the disruption generator, instance lists, random seeds, raw result files, and the figure-generation scripts, is archived on Zenodo (DOI: 10.5281/zenodo.20631104) and maintained at https://github.com/MadBezoui/reactive_MO_scheduling.

Author contributions. L.O.: conceptualisation, methodology, software, investigation, writing, original draft. M.B.: methodology, formal analysis, validation, writing, review and editing. K.A.: supervision, writing, review and editing. All authors read and approved the final manuscript.

Ethics approval. Not applicable.

Appendix A Historical ablation not used for final claims

This appendix documents an earlier four-term prototype for transparency; its results are not used to support the claims of the paper because they relied on a non-normalised three-coordinate hypervolume proxy. The prototype combined the scope-sharing weight with the branching-history term HD retained in the main text, a multi-objective structural term (MO) and a perturbation-awareness term (DYN). A leave-one-out ablation on the three tuning instances (`la06-la08`, $n = 15$ paired), computed on that proxy, reported HV ratios vs `dom/wdeg` of 1.381 (all terms), 1.291 (remove HD), 1.379 (remove MO), 1.377 (remove DYN), 1.404 (remove the scope-sharing core): HD appeared to "contribute most" while MO and DYN were inert ($< 0.5\%$). Because the proxy folds in a flexibility axis and uses only three tuning instances, it cannot support a claim about a single term; Section 6.2 shows that on the final normalised two-objective hypervolume HD is HV-neutral. MO and DYN had no measurable effect and were removed, so the shipped selector reduces exactly to `dom/wdeg2004`+HD (Eq. (10)).

Appendix B Additional sensitivity

We report four robustness checks: the reference-point check (A) uses the full benchmark; the harness-based checks (B)-(D) run on a small representative set of RCPSP instances and are sanity checks rather than an exhaustive sensitivity study.

(A) Hypervolume reference point.

Table B2 reports the Friedman mean ranks on the normalised 2D hypervolume for $r \in \{1.05, 1.1, 1.2\}$; the ordering is invariant.

(B) ε-grid size κ.

On the instrumented harness (`dom/wdeg2004`, 3 s/ε-point), varying κ leaves closure essentially unchanged while node count and CPU scale predictably with the number of ε-points (Table B1).

(C) Time budget.

Increasing the per-ε-point limit changes absolute closure (the ε-grid is recomputed from budget-dependent bounds, so closure need not be monotone on a single instance), but the relative relationship between the selectors is unchanged; the main campaign already reports the 30 s budget and the controlled diagnostic the 3 s budget with consistent rankings.

Table B1 ε-grid-size sensitivity (`dom/wdeg2004`, 3 s per ε-point).

Instance	Metric	κ=3	κ=5	κ=10
`j3010_1`	closure	0.80	0.86	0.83
	nodes	148,092	197,354	430,972
	CPU (s)	7.9	10.9	20.8
`j3012_1`	closure	1.00	1.00	1.00
	nodes	47	49	54
	CPU (s)	0.03	0.04	0.04

(D) HD weight α.

Sweeping $\alpha \in \{0.1, 0.3, 0.5, 0.7\}$ for dom/wdeg2004+HD on RCPSP leaves closure invariant (e.g. `j3010_1` at 0.50 and `j3011_1` at 0.75 for every α) and moves node counts only within run-to-run noise, consistent with HD being a second-order, search-effort term.

Table B2 Friedman mean rank on normalised HV vs reference point r; ordering stable.

Family	Selector	r=1.05	r=1.1	r=1.2
Job-shop	NSGA-III	1.48	1.48	1.48
	Choco `dom/wdeg`	3.59	3.60	3.59
	dom/wdeg2004+HD	3.74	3.71	3.70
	`activity`	5.77	5.77	5.77
RCPSP	`dom`	2.33	2.34	2.34
	Choco `dom/wdeg` / +HD	2.88	2.89	2.89
	`activity`	4.35	4.31	4.31

References

Aribi D, Belkahla Driss O, El Haouzi H (2023) Multi-objective optimization of the dynamic and flexible job shop scheduling problem under workers fatigue constraints. In: Proceedings of the 15th International Conference on Agents and Artificial Intelligence. SCITEPRESS - Science and Technology Publications, pp 301–308, https://doi.org/10.5220/0011663800003393

Bär S (2022) Multi-Agent Approach for Reactive Scheduling in Flexible Manufacturing, Springer Fachmedien Wiesbaden, pp 75–97. https://doi.org/10.1007/978-3-658-39179-9_5

Bezoui M, Kermali A, Bounceur A, et al (2023a) Deep reinforcement learning for multiobjective scheduling in industry 5.0 reconfigurable manufacturing systems.

In: International Conference on Machine Learning for Networking (MLN 2023). Springer Nature Switzerland, Cham, pp 90–107

Bezoui M, Olteanu AL, Sevaux M (2023b) Integrating preferences within multiobjective flexible job shop scheduling. European Journal of Operational Research 305(3):1079–1086

Boussemart F, Hemery F, Lecoutre C, et al (2004) Boosting systematic search by weighting constraints. In: Proceedings of the 16th European Conference on Artificial Intelligence (ECAI'04). IOS Press, pp 146–150

Chang J, Yu D, Hu Y, et al (2022) Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival. Processes 10(4):760. https://doi.org/10.3390/pr10040760

Da Col G, Teppan EC (2022) Industrial-size job shop scheduling with constraint programming. Operations Research Perspectives 9:100–249. https://doi.org/10.1016/j.orp.2022.100249

Deb K, Jain H (2014) An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Transactions on Evolutionary Computation 18(4):577–601

Deb K, Pratap A, Agarwal S, et al (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2):182–197

Duan C, Mo Y, Zhang Z (2025) Deep reinforcement learning for event-driven predictive–reactive multi-objective scheduling in dynamic flexible job shop. Journal of Manufacturing Systems 83:992–1008. https://doi.org/10.1016/j.jmsy.2025.11.014

Ebufegha A, Li S (2022) Multi-agent system model for dynamic scheduling in flexible job shop subject to random machine breakdown. In: 2022 Winter Simulation Conference (WSC). IEEE, pp 1719–1730, https://doi.org/10.1109/wsc57314.2022.10015488

Geibinger T, Mischek F, Musliu N (2024) Investigating constraint programming and hybrid methods for real world industrial test laboratory scheduling. Journal of Scheduling 27(6):607–622. https://doi.org/10.1007/s10951-024-00821-0

Grimes D, Hébrard E (2015) Solving variants of the job shop scheduling problem through conflict-directed search. INFORMS Journal on Computing 27(2):268–284. https://doi.org/10.1287/ijoc.2014.0625

Habet D, Terrioux C (2019) Conflict history based search for constraint satisfaction problem. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing (SAC'19), Knowledge Representation and Reasoning Track. ACM, pp

1117–1122, https://doi.org/10.1145/3297280.3297389

Haimes YY, Lasdon LS, Wismer DA (1971) On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Transactions on Systems, Man, and Cybernetics SMC-1(3):296–297

Hébrard E, Siala M (2017) Explanation-based weighted degree. In: Integration of AI and OR Techniques in Constraint Programming (CPAIOR 2017), Lecture Notes in Computer Science, vol 10335. Springer, pp 167–175, https://doi.org/10.1007/978-3-319-59776-8_13

Herroelen W, Leus R (2005) Project scheduling under uncertainty: Survey and research potentials. European Journal of Operational Research 165(2):289–306

Kolisch R, Sprecher A (1997) PSPLIB – a project scheduling problem library. European Journal of Operational Research 96(1):205–216

Laborie P, Rogerie J, Shaw P, et al (2018) IBM ILOG CP optimizer for scheduling: 20+ years of scheduling with constraints at IBM/ILOG. Constraints 23(2):210–250

Lan L, Berkhout J (2025) PyJobShop: Solving scheduling problems with constraint programming in Python. ArXiv preprint arXiv:2502.13483

Maenhout B, Vanhoucke M (2024) Dynamic personnel rescheduling: insights and recovery strategies. Journal of Scheduling 27(1):1–27. https://doi.org/10.1007/s10951-023-00785-7

Michel L, Van Hentenryck P (2012) Activity-based search for black-box constraint programming solvers. In: Integration of AI and OR Techniques in Constraint Programming (CPAIOR). Springer, pp 228–243

Muth JF, Thompson GL (1963) Industrial scheduling. Prentice-Hall, Englewood Cliffs, NJ

Ouelhadj D, Petrovic S (2009) A survey of dynamic scheduling in manufacturing systems. Journal of Scheduling 12(4):417–431

Pedrosa J, Puig V (2023) Job shop scheduling with limited-capacity buffers using constraint programming and genetic algorithms. IFAC-PapersOnLine 56(2):953–958. https://doi.org/10.1016/j.ifacol.2023.10.1688

Pleier F, Schilp J (2024) Predictive-reactive scheduling to increase robustness of production systems. In: 2024 4th International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). IEEE, pp 1–6, https://doi.org/10.1109/iceccme62383.2024.10796054

Prud'homme C, Fages JG (2022) Choco-solver: A Java library for constraint programming. Journal of Open Source Software 7(78):4708. https://doi.org/10.21105/joss.

04708

Rener E, Salassa F, T'kindt V (2025) Single machine rescheduling for new orders with maximum and total time disruption constraints. Journal of Scheduling 28(5):469–482. https://doi.org/10.1007/s10951-025-00837-0

Sáez Bustos P, Parada V (2025) A reactive hybrid product-driven system for rescheduling in a manufacturing planning. Brazilian Journal of Operations & Production Management 22(2):2570. https://doi.org/10.14488/BJOPM.2570.2025

Teppan E (2022) Types of flexible job shop scheduling: A constraint programming experiment. In: Proceedings of the 14th International Conference on Agents and Artificial Intelligence. SCITEPRESS - Science and Technology Publications, pp 516–523, https://doi.org/10.5220/0010849900003116

Vieira GE, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: A framework of strategies, policies, and methods. Journal of Scheduling 6(1):39–62

Wattez H, Lecoutre C, Paparrizou A, et al (2019) Refining constraint weighting. In: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, pp 71–77, https://doi.org/10.1109/ICTAI.2019.00019

Wu R, Zheng J, Yin X (2025) Dynamic scheduling for multi-objective flexible job shops with machine breakdown by deep reinforcement learning. Processes 13(4):1246. https://doi.org/10.3390/pr13041246

Xu H, Zheng J, Huang L, et al (2025) Solving dynamic multi-objective flexible job shop scheduling problems using a dual-level integrated deep Q-Network approach. Processes 13(2):386. https://doi.org/10.3390/pr13020386

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation 3(4):257–271