Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 23, No. 2, 2026

Risk-aware Balancing of Cooperative Mixed-model Assembly Lines with Correlated Task Times

Yuzhen Wang[1, *]

[1]Shandong University of Science and Technology, Qingdao, China
[*]Corresponding author: 202483010196@sdust.edu.cn

Abstract: This study jointly optimizes task assignments, worker crews and sequences in cooperative mixed-model assembly lines with correlated task times. A lognormal common-factor model represents within-product dependence, and a tail-bottleneck-guided adaptive neighborhood search minimizes the conditional value-at-risk of normalized cycle-time overruns under fixed staffing. Crews and orders are committed before uncertainty is realized; task starts depend on predecessor completion. Seven methods are evaluated through twenty runs on each of 48 instances extended from eight public precedence graphs. Complete enumeration and an independent SimPy model verify small-instance solutions and execution consistency. At 2000 candidate evaluations, ALNS-R reduces mean out-of-sample CVaR by 20.70% relative to deterministic balancing. Product-stream CVaR decreases by 16.08%, while normalized throughput increases by 0.56% in relative terms. Correlation and budget analyses delimit these gains, including nonsignificant comparisons and nonmonotonic scenario-count effects.

Keywords: Mixed-model assembly line balancing; cooperative tasks; correlated task times; conditional value-at-risk; adaptive neighborhood search; discrete-event simulation.

1. Introduction

Mixed-model assembly line balancing allocates tasks while respecting technological precedence and station resources; established classifications and methods are reviewed in [1-3]. In cooperative operations, participating workers must start and finish a task together. Station completion therefore depends on resource contention and synchronization as well as total labor content.

Previous studies provide complementary foundations for these decisions. Two-sided U-shaped and mixed-model parallel two-sided layouts are addressed in [4,5], while collaborative mixed-model balancing using deep reinforcement learning is examined in [6]. Other contributions consider multi-manned balancing and within-station scheduling [7,8], or mixed-model lines with walking workers [9]. These formulations motivate a representation that specifies both task allocation and the workers available for each operation.

Processing-time uncertainty further couples balancing and execution. Relevant approaches include stochastic multi-manned balancing [10], chance constraints [11], line-level reliability methods [12], robust optimization [13] and simulation-based rebalancing [14]. Here the concern is the interaction between correlated processing times and simultaneous worker occupation under an explicit synchronous transfer rule. A common product-level disturbance may affect several assigned tasks, while a cooperative task also waits until every crew member is free.

Building on these formulations, this study combines correlated times, simultaneous crew occupation and line-level tail risk in an executable scheduling model with crews and task orders fixed in advance. A bottleneck-guided search is evaluated against six alternatives, while continuous product streams assess whether improvements in sampled tail risk translate into more reliable cycles and higher average throughput.

2. Risk Model

The line contains K sequential stations and three product models. Each station has four fixed workers, two in each of two skill categories. Task i requires skill e_i and $p_i \in \{1,2\}$ workers; preemption, splitting and mid-task replacement are prohibited. A two-worker task lasting t consumes $2t$ labor units. There are no intermediate buffers; handling, setup and walking times are ignored. Products enter independently according to the specified mix. Different products are independent, while tasks in the same process group of one product share a disturbance. Every method uses the same K stations and $4K$ workers, including idle workers.

A solution $A = (\boldsymbol{a}, G, \pi)$ specifies station assignments a_i, fixed crews G_i and a global topological order π. Let $W(k, e)$ be the workers of skill e at station k. Crews and relative resource orders remain fixed across models and scenarios. Cross-station predecessors are completed upstream before a product arrives. The set $H_i(A)$ contains task i's direct technological predecessors at its station and each crew member's preceding task in π. Start, finish and station completion times follow

$$\begin{aligned} s_i(A,\omega) &= \max\big(\{0\} \cup \{f_j(A,\omega): j \in H_i(A)\}\big), \\ f_i(A,\omega) &= s_i(A,\omega) + t_{im}(\omega). \end{aligned} \tag{1}$$

$$C_k(A,\omega) = \max(\{0\} \cup \{f_i(A,\omega): a_i = k\}). \tag{2}$$

Since every predecessor occurs earlier in π, decoding is acyclic. Shared workers execute tasks sequentially, cooperative crews start and finish together, and starts depend only on completed predecessors. This policy is non-anticipative. For the precedence edge set E and product-model set M, feasibility requires

$$\begin{aligned} a_i &\in \{1, \dots, K\}, \\ a_i \le a_j, \quad \pi(i) < \pi(j), &\qquad \forall (i,j) \in E. \end{aligned} \tag{3}$$

$$G_i \subseteq W(a_i, e_i), \qquad |G_i| = p_i,$$
$$\max_{m \in M} C_k(A, \boldsymbol{\mu}_m) \le T, \qquad \forall k \in \{1, \dots, K\}. \tag{4}$$

Thus, precedence, skills, crew size and nominal cycle time are enforced; stochastic overruns enter the objective. Mean processing times are μ_{im}, and the coefficient of variation is v. For a fixed process group $g(i)$, independent standard normal group factors Z_g and individual factors ϵ_i give

$$t_{im} = \mu_{im} \exp\left\{\tau\left[\sqrt{\rho}\, Z_{g(i)} + \sqrt{1-\rho}\, \epsilon_i\right] - \frac{\tau^2}{2}\right\}, \qquad \tau = \sqrt{\ln(1+v^2)}. \tag{5}$$

Conditional on the model, within-group correlation is $[\exp(\rho\tau^2) - 1]/[\exp(\tau^2) - 1]$, and between-group correlation is zero. The latent-normal parameter ρ changes dependence without changing marginal distributions; model mixing can add unconditional dependence.

A scenario ω contains the models and task durations of the distinct products occupying the stations. All stations transfer synchronously after their work is complete and the target T has elapsed. Transfer period P and normalized overrun loss L are

$$P(A, \omega) = \max\{T, C_1(A, \omega), \dots, C_K(A, \omega)\}, \qquad L(A, \omega) = \frac{P(A,\omega) - T}{T}. \tag{6}$$

Using the CVaR representation [15], with auxiliary threshold η and excess losses ξ_s, the population objective and its approximation with S scenarios are

$$\min_A \mathrm{CVaR}_\alpha(L) = \min_{A,\eta}\left\{\eta + \frac{\mathrm{E}[(L-\eta)_+]}{1-\alpha}\right\} \tag{7}$$

$$\min_{A,\eta,\xi} \quad \eta + \frac{1}{(1-\alpha)S}\sum_{s=1}^{S} \xi_s, \qquad \text{s.t.} \quad \xi_s \ge L(A, \omega_s) - \eta, \quad \xi_s \ge 0, \qquad s = 1, \dots, S. \tag{8}$$

Optimization satisfies Eqs. (3)-(4). With $\alpha = 0.90$ and S a multiple of ten, sample CVaR equals the mean of the largest 10% of sampled losses.

3. Adaptive Search

An initial topological sequence selects eligible tasks by positional weight: the model-mix-weighted mean time of the task plus all successors. Stations are divided by cumulative labor time, single-worker tasks alternate between eligible workers, and nominal search provides a shared feasible start. Four operators, following the adaptive-search framework [16], relocate a task, switch a single worker, swap adjacent tasks without a precedence edge, or perturb two to four tasks around a selected station. Relocation ranges are updated from immediate predecessor and successor station indices. Focused perturbation also switches a single-worker assignment with probability 0.5; nominal feasibility is checked before scenario evaluation.

For scenarios in the largest-loss decile Tail, the station with maximum completion time is recorded, with ties resolved by the smallest station index. Additive smoothing yields the sampling probabilities

$$b_k = \frac{0.01 + \sum_{s \in \mathrm{Tail}} 1\left\{k = \arg\max_j C_j(A, \omega_s)\right\}}{0.01K + |\mathrm{Tail}|} \tag{9}$$

After station selection, at most ten attempts locate a task assigned there; otherwise a random task is retained. Operator weights start at one and are updated every 100 evaluations as $w_q \leftarrow 0.8 w_q + 0.2[0.1 + h_q/\max(1, c_q)]$, where c_q is usage and h_q scores five points for a new overall best or one for improvement only over the current solution. Counts reset after updating. Annealing temperature decreases exponentially from 0.04 to 0.001, with a return to the best solution every 400 evaluations.

ALNS-R uses Eq. (9); ALNS-U samples stations uniformly; SA uses only the first three single-move neighborhoods. In-sample scenarios remain fixed, and solutions are selected without test results. Infeasible candidates count toward $N_{\max}$; initial and restart reevaluations are separate.

Let $n = |I|$ be the task count and d the mean number of within-station direct predecessors. Structural checks take $O(n^2)$ time. Scenario recursion adds $O(Sn(d+2) + SK)$, and sorting losses takes $O(S\log S)$. Their sum bounds one complete candidate evaluation.

Bottleneck frequencies guide the offline search rather than task dispatch during production. Once a solution is chosen, neither its crews nor its resource order can react to a future duration. Fixed training scenarios make candidate comparisons consistent within a run; an independent test set assesses the selected solution after search. The algorithm updates the best solution whenever its training objective improves and returns that solution at the candidate limit, including when the final current solution is worse.

4. Computational Experiments

Design and verification. Eight public graphs [17] are used: MITCHELL (21 tasks), GUNTHER (35), HAHN (53), TONGE70 (70), ARC83 (83), LUTZ3 (89), ARC111 (111) and BARTHOLD (148). Three seeds and cooperative proportions $p = 0.2, 0.4$ give 48 instances. Model-specific times, skills, crew sizes and process groups are synthetic extensions. Base times are scaled to mean ten; models 2 and 3 multiply model-1 times by independent uniform factors on [0.8,1.2]. Mix probabilities are 0.5, 0.3 and 0.2. Two skills and four process groups are assigned uniformly. The first round(pn) tasks in a random permutation require two workers; paired proportions share all remaining attributes.

Set $K = \lceil n/8 \rceil$ and T to 1.03 times the maximum model completion time of the best solution from 8000 nominal evaluations. Main settings are $v = 0.20$, $\rho = 0.50$, $S = 300$ and 10000 independent test scenarios. DET minimizes maximum nominal completion time; IND minimizes CVaR assuming $\rho = 0$; MEAN minimizes expected loss under correlation; SA, ALNS-U, ALNS-R and GA minimize correlated-time CVaR. Each method runs twenty times per instance with $N_{\max} = 2000$, giving 6720 searches. All share resources, starting solutions and a fixed seed protocol. Python 3.12 and Numba implement the methods.

GA uses 24 individuals, three-way tournaments, crossover probability 0.8, precedence and station repair, one to three mutation moves, and worst-member replacement with elitism. Population evaluations beyond the shared initial solution count toward the budget. On four two-station BOWMAN8 cases with two cooperative tasks, complete enumeration

covers $16 \times 16 \times 64 = 16384$ representations per case. Exact 300-scenario CVaRs are 0.153379, 0.197417, 0.226884 and 0.091315; ALNS-R and GA each reach these optima in 20/20 runs. Independent SimPy 4.1.2 validation [18] checks DET and ALNS-R on eight representative instances, with 500 products after $K - 1$ filling cycles per trajectory. All sixteen trajectories agree with recursive transfer periods to floating-point precision. Fig. 1 illustrates simultaneous crew occupation.

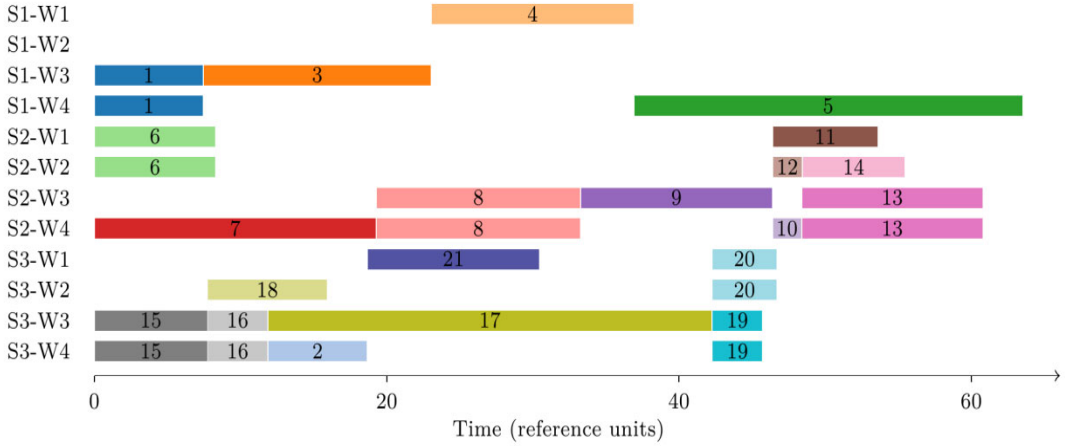


Figure 1. Resource occupancy for a MITCHELL instance

Risk comparisons. Table 1 reports equal-weight averages across all instances. Combining the two proportions for each graph and seed gives 24 groups. Two-sided paired Wilcoxon tests use Holm correction for six comparisons against ALNS-R; graph-level tests additionally average the three seeds within each graph family.

Table 1. Out-of-sample risk and grouped comparisons

Method	CVaR	Mean loss	Adjusted p (24 groups)	Adjusted p (8 graphs)
DET	0.1788	0.0337	$<10^{-4}$	0.0469
IND	0.1432	0.0240	0.1875	0.5469
MEAN	0.1513	0.0241	0.0210	0.2969
SA	0.1735	0.0303	$<10^{-4}$	0.0938
ALNS-U	0.1506	0.0247	0.0117	0.1172
ALNS-R	0.1418	0.0232	–	–
GA	0.1692	0.0280	$<10^{-4}$	0.0781

ALNS-R reduces CVaR by 20.70% relative to DET. The 24-group comparisons support improvements over DET, MEAN, SA, ALNS-U and GA, but not IND. Only DET remains significant at the eight-graph level. Grouped bootstrap 95% intervals for DET-minus-ALNS-R and ALNS-U-minus-ALNS-R CVaR differences are [0.0277,0.0467] and [0.0037,0.0154]. Thus, risk differences and the limited graph-level evidence must be interpreted together.

The adjusted IND comparison has $p = 0.1875$, whereas the graph-level DET comparison has $p = 0.0469$. The former does not establish a statistically detectable benefit of explicitly modeling correlation in the primary setting. Similarly, the nonsignificant graph-level comparisons cannot support general superiority across precedence structures. The positive bootstrap intervals quantify estimated differences under the 24-group resampling scheme; they do not remove the limitations of the smaller graph-family analysis.

Sensitivity. Correlation tests use the 40%-cooperative MITCHELL, TONGE70, ARC111 and BARTHOLD instances: three correlation levels, three methods and ten runs at 10000 evaluations give 360 searches. IND always optimizes at zero correlation and matches ALNS-R when $\rho = 0$. At $\rho = 0.8$, ALNS-R's mean advantage over IND is 1.64% (Table 2); this four-instance result does not establish a general trend.

Table 2. Sensitivity analysis of the correlation parameter

ρ	IND CVaR	MEAN CVaR	ALNS-R CVaR
0.0	0.1287	0.1307	0.1287
0.5	0.1523	0.1547	0.1520
0.8	0.1683	0.1679	0.1655

Budget tests use seed-zero, 40%-cooperative instances of all eight graphs and ten runs per setting (Table 3). The 720 records reuse 240 main-experiment runs, requiring 480 additional searches. Relative ALNS-R improvements over ALNS-U are 4.34%, 0.32% and 2.20%; these are candidate-budget comparisons, not wall-clock speed or convergence guarantees. For the four correlation-test graphs, $S = 100,300,1000$ at 10000 evaluations and ten runs gives 120

searches, evaluated on 20000 scenarios. Mean CVaRs are 0.1570, 0.1503 and 0.1553; the scenario-count effect is nonmonotonic.

Table 3. Candidate-evaluation limits and out-of-sample risk

$N_{\max}$	ALNS-U CVaR	ALNS-R CVaR	GA CVaR
2000	0.1553	0.1486	0.1890
10000	0.1309	0.1305	0.1711
50000	0.1204	0.1178	0.1554

Continuous product streams. Each instance's seven best in-sample solutions receive ten paired streams of 20000 completed products after $K-1$ filling cycles: 480 shared inputs generate 3360 method trajectories. Product models and full task-time vectors persist across stations; in cycle r, station k handles product $r-k+1$. The SimPy-validated recursion computes each period. Stream-specific statistics are averaged within instances, then equally across instances. Normalized throughput is T divided by each stream's mean period.

Table 4. Product-stream simulation results

Method	Stream CVaR	Overrun rate / %	Normalized throughput
DET	0.1397	26.05	0.9764
IND	0.1186	20.93	0.9812
MEAN	0.1202	19.57	0.9822
SA	0.1323	22.82	0.9796
ALNS-U	0.1193	20.04	0.9816
ALNS-R	0.1173	19.68	0.9818
GA	0.1412	23.83	0.9780

Relative to DET, ALNS-R lowers stream CVaR by 16.08% (Holm-adjusted $p < 10^{-4}$), overrun frequency by approximately 6.4 percentage points, and raises normalized throughput by 0.56% in relative terms. Since periods cannot fall below T, normalized throughput cannot exceed one. MEAN gives slightly higher throughput but greater tail risk. Main results summarize all search runs, whereas stream results evaluate selected solutions.

5. Conclusions

Under fixed staffing, the proposed model and bottleneck-guided search reduce severe overruns relative to deterministic balancing. Improvements over uniform guidance depend on the evaluation budget, and correlation effects remain descriptive on four instances. Product-stream risk and throughput gains have different magnitudes. These findings concern synthetic extensions of public graphs, zero buffers and equal worker abilities within skills. Future work should calibrate dependence using industrial times and examine finite buffers, heterogeneous workers and failures.

For line design, CVaR complements rather than replaces the mean output measure: reducing severe overruns need not yield a proportional throughput gain because the target cycle time imposes a lower bound on each transfer period. The experiments therefore support reporting both indicators for a fixed staffing configuration. They do not estimate savings for a particular factory, whose time distributions and dependence structure would first require calibration.

References

[1] Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. European Journal of Operational Research, 183(2), 674–693. https://doi.org/10.1016/j.ejor.2006.10.010

[2] Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics, 142(2), 259–277. https://doi.org/10.1016/j.ijpe.2012.10.020

[3] Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 168(3), 666–693. https://doi.org/10.1016/j.ejor.2004.07.022

[4] Jiao, Y. L., Deng, X., Li, L., et al. (2023). Balancing and collaborative optimization of two-sided U-type assembly line with multi-constraint. Journal of Jilin University (Engineering and Technology Edition), 53(7), 2053–2060. https://doi.org/10.13229/j.cnki.jdxbgxb.20211050

[5] Jiao, Y. L., Deng, X., Li, J., et al. (2025). Mathematical model and heuristics for balancing of mixed-model parallel two-sided assembly line. Computer Integrated Manufacturing Systems, 31(8), 2797–2805. https://doi.org/10.13196/j.cims.2022.0995

[6] Zhang, M., Tian, Z. Y., Zhu, J. H., et al. (2024). Multi-manned collaborative mixed-model assembly line balancing optimization based on deep reinforcement learning. Control and Decision, 39(10), 3395–3404. https://doi.org/10.13195/j.kzyjc.2023.0820

[7] Roshani, A., & Ghazi Nezami, F. (2017). Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach. Assembly Automation, 37(1), 34–50. https://doi.org/10.1108/aa-02-2016-016

[8] Michels, A. S., & Costa, A. M. (2024). Model and heuristics for the multi-manned assembly line worker integration and balancing problem. International Journal of Production Research, 62(24), 8719–8744. https://doi.org/10.1080/00207543.2024.2347572

[9] Hashemi-Petroodi, S. E., Thevenin, S., Kovalev, S., et al. (2023). Markov decision process for multi-manned mixed-model assembly lines with walking workers. International Journal of Production Economics, 255, 108661. https://doi.org/10.1016/j.ijpe.2022.108661

[10] Zakaraia, M., Zaher, H., & Ragaa, N. (2022). Solving stochastic multi-manned U-shaped assembly line balancing problem using differential evolution algorithm. International

Journal of Production Management and Engineering, 10(1), 13–22. https://doi.org/10.4995/ijpme.2021.16084

[11] Yasmeen, S., Keskin, M. E., & Yilmaz, H. (2025). Multi-manned assembly line balancing problem with stochastic processing times. Flexible Services and Manufacturing Journal. https://doi.org/10.1007/s10696-025-09641-6

[12] Diefenbach, J., & Stolletz, R. (2022). Stochastic assembly line balancing: General bounds and reliability-based branch-and-bound algorithm. European Journal of Operational Research, 302(2), 589–605. https://doi.org/10.1016/j.ejor.2022.01.015

[13] Samouei, P., & Ashayeri, J. (2019). Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations. Applied Mathematical Modelling, 72, 259–275. https://doi.org/10.1016/j.apm.2019.02.019

[14] Nikkerdar, M., ElMaraghy, W., & ElMaraghy, H. (2025). Smart adaptable assembly line rebalancing based on reinforcement learning. The International Journal of Advanced Manufacturing Technology, 138(9–10), 4155–4173. https://doi.org/10.1007/s00170-025-15780-9

[15] Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. The Journal of Risk, 2(3), 21–41. https://doi.org/10.21314/jor.2000.038

[16] Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40(4), 455–472. https://doi.org/10.1287/trsc.1050.0135

[17] Scholl, A. (1993). Benchmark data sets by Scholl (1993). https://assembly-line-balancing.de/salbp/benchmark-data-sets-1993/

[18] SimPy Developers. (n.d.). SimPy documentation: Overview. https://simpy.readthedocs.io/en/latest/