JURNAL INDUSTRI & INOVASI
Jurnal INVASI
ISSN: ISSN 3046 -8019
PUBLISHER: Program Studi Teknik Industri
Fakultas Teknik, Universitas Teuku Umar
Aceh Barat, Indonesia

Studi Komparatif MILP dan Heuristik pada Job Shop Scheduling Problem: Minimasi Makespan dan Total Tardiness

Andrianyah Hamid*[1]
Program Studi Teknik Industri, Fakultas Teknik, Universitas Syiah Kuala
*Email Korespondensi: andriansyah@usk.ac.id

Abstract *- This study presents a comprehensive analysis of the Job Shop Scheduling Problem (JSSP) with two commonly used objectives in manufacturing: minimizing makespan and minimizing total tardiness. Two approaches are directly compared: an exact model based on Mixed Integer Linear Programming (MILP) solved using Gurobi, and a fast heuristic based on dispatching rules and local search. The study is conducted on 10 random instances (10 jobs, 5 machines, 5 operations per job). The results show that the heuristic is highly competitive for the makespan objective (average gap of 9.25% with approximately 240 times faster computation than MILP). For the tardiness objective under a balanced configuration, the heuristic yields an average relative gap (measured only on instances where the MILP objective value is positive, i.e., conditional average gap) of 52.39% with a runtime cost about 6.0 times slower than MILP. This paper provides a reproducible baseline, data-driven analysis, and technical directions for improving both the quality and efficiency of tardiness-focused heuristics in future research.*

Keywords: *Job Shop Scheduling, MILP, heuristic, makespan, tardiness*

PENDAHULUAN

JSSP merupakan salah satu masalah optimasi kombinatorial klasik yang tetap relevan pada sistem produksi modern (Garey, Johnson, & Sethi, 1976; Manne, 1960). Setiap job memiliki urutan operasi yang tetap, dan setiap operasi hanya dapat dikerjakan pada mesin tertentu. Kompleksitas masalah timbul dari kombinasi dua jenis kendala utama: (i) *precedence* antar operasi dalam job yang sama dan (ii) konflik kapasitas pada mesin.

Dalam praktik, objektif penjadwalan tidak tunggal. Objektif makespan menekankan efisiensi total horizon produksi, sedangkan objektif total tardiness lebih berorientasi pada kinerja pemenuhan *due date*. Keduanya dapat menghasilkan karakter keputusan yang berbeda (Pinedo, 2016). Dalam satu dekade terakhir, pengembangan metode eksak untuk masalah *job shop* terus berlanjut melalui evaluasi komputasional model *Mixed Integer Programming* (MIP) dan *Constraint Programming* (CP) untuk masalah penjadwalan dengan banyak kendala (Ku & Beck, 2016; Laborie, Rogerie, Shaw, & Vilím, 2018). Di sisi lain, lingkungan produksi yang dinamis mendorong kemunculan pendekatan adaptif berbasis pembelajaran (*learning)*, khususnya *deep reinforcement learning* untuk *rescheduling* dan *dispatching* secara *real-time* (Gui, Tang, Zhu, Zhang, & Zhang, 2023; Lv, Zhang, Fan, & Shen, 2025; Wang et al., 2021).

Berdasarkan konteks tersebut, naskah ini berfokus pada tiga pertanyaan penelitian yang dirumuskan secara terpadu: seberapa dekat kualitas heuristik terhadap solusi MILP untuk objektif makespan dan tardiness, seberapa besar penghematan runtime yang dapat diperoleh heuristik terhadap pendekatan eksak, serta pada objektif mana heuristik sederhana masih memadai dan pada objektif mana dibutuhkan penguatan metode.

Kontribusi penelitian ini dinyatakan melalui empat aspek utama yang saling terkait, yaitu penyusunan *baseline* eksperimental JSSP yang reprodusibel untuk dua objektif sekaligus, perumusan dan implementasi model MILP yang konsisten untuk makespan dan tardiness, pengembangan heuristik tardiness yang diperkuat melalui dispatch menggunakan *Apparent Tardiness Cost* (ATC), pencarian lokal berbasis insertion, swap, multi-start, dan iterated local search (Vepsalainen & Morton, 1987), serta analisis komparatif kuantitatif berbasis 10 instance yang didukung visualisasi informatif. Integrasi sistematis ATC multi-start dengan iterated local search dalam kerangka dua fase (diversifikasi-intensifikasi) merupakan novelty teknis utama studi ini, membedakannya dari implementasi dispatch rule konvensional yang umumnya hanya menggunakan satu titik awal tanpa mekanisme perturbasi terarah.

TINJAUAN PUSTAKA

Literatur JSSP menunjukkan dua arus utama: pendekatan eksak untuk kualitas solusi tinggi pada skala terbatas, dan pendekatan heuristik/metaheuristik untuk efisiensi komputasi pada skala yang lebih besar (Jain & Meeran, 1999; Pinedo, 2016). Formulasi disjunktif untuk *job shop* telah lama menjadi rujukan dasar dalam pemodelan eksak (Manne, 1960) Untuk praktik industri, heuristik berbasis aturan dispatch tetap populer karena implementasinya cepat, transparan, dan andal (Jain & Meeran, 1999).

Survei terbaru menunjukkan bahwa diversifikasi model JSSP pada periode modern semakin tinggi, termasuk dimensi dinamis, fleksibilitas mesin, dan perluasan struktur kendala yang lebih realistis (Xiong, Shi, Ren, & Hu, 2022). Dalam konteks eksak, kombinasi MILP dan CP juga digunakan pada varian sistem produksi dinamis untuk meningkatkan kualitas keputusan penjadwalan (Fatemi-Anaraki, Tavakkoli-Moghaddam, Foumani, & Vahedi-Nouri, 2023). Sementara itu, pada objektif yang terkait *due date*, indikator prioritas seperti *urgency/slack* lazim digunakan karena lebih sensitif terhadap risiko keterlambatan dibandingkan aturan yang semata berbasis waktu proses (Vepsalainen & Morton, 1987). Tren terbaru juga menunjukkan peningkatan signifikan riset *deep reinforcement learning* untuk penjadwalan job shop dinamis (Gui et al., 2023; Lv et al., 2025; Wang et al., 2021). Secara komparatif, heuristik berbasis dispatch rule sederhana umumnya menghasilkan gap 10–40% terhadap solusi eksak pada benchmark berukuran moderat, sedangkan metaheuristik seperti genetic algorithm, tabu search, dan simulated annealing dapat menekan gap hingga di bawah 5% namun dengan biaya komputasi yang jauh lebih tinggi dan parameter tuning yang lebih kompleks (Jain & Meeran, 1999; Xiong et al., 2022). Evaluasi langsung antara dispatch rule dan MILP pada instance berbasis due date sintetis seperti yang dilakukan studi ini, masih relatif jarang dalam literatur. Studi ini mengambil posisi sebagai baseline engineering: bukan mengklaim algoritma *state-of-the-art*, tetapi menegaskan trade-off kualitas-waktu secara terukur.

FORMULASI MILP

Bagian ini merumuskan model MILP JSSP secara terstruktur, dimulai dari notasi, fungsi objektif, kumpulan kendala, hingga domain variabel keputusan.

Tabel 1 Notasi

Simbol	Keterangan
$\mathcal{J}$	himpunan job, dengan indeks j.
$\mathcal{M}$	himpunan mesin.
$\mathcal{O}$	himpunan seluruh operasi.
$\mathcal{O}_j$	himpunan operasi milik job j.
$\mathcal{A}$	himpunan arc precedence antar-operasi dalam job yang sama.
$\mathcal{P}$	himpunan pasangan operasi pada mesin yang sama.
p_u	waktu proses operasi $u \in \mathcal{O}$.
d_j	due date job j.

Simbol	Keterangan
u_j^{last}	operasi terakhir pada job j.
M	konstanta besar untuk linearisasi kendala disjunktif.
S_u	variabel keputusan waktu mulai operasi u.
$C_{\max}$	variabel makespan.
C_j	variabel waktu selesai job j.
T_j	variabel tardiness job j.
y_{uv}	variabel biner urutan operasi untuk pasangan $(u, v) \in \mathcal{P}$.

$$\min Z_1 = C_{\max} \tag{1}$$

$$\min Z_2 = \sum_{j \in \mathcal{J}} T_j \tag{2}$$

$$S_v \geq S_u + p_u, \quad \forall (u \to v) \in \mathcal{A}, \tag{3}$$

$$S_v \geq S_u + p_u, \quad \forall (u \to v) \in \mathcal{A}, \tag{4}$$

$$S_v \geq S_u + p_u, \quad \forall (u \to v) \in \mathcal{A}, \tag{5}$$

$$C_{\max} \geq S_u + p_u, \quad \forall u \in \mathcal{O}, \tag{6}$$

$$C_j = S_{u_j^{\text{last}}} + p_{u_j^{\text{last}}}, \quad \forall j \in \mathcal{J}, \tag{7}$$

$$T_j \geq C_j - d_j, \quad \forall j \in \mathcal{J}. \tag{8}$$

$$S_u \geq 0,\ C_{\max} \geq 0,\ C_j \geq 0,\ T_j \geq 0,\ y_{uv} \in \{0,1\}. \tag{9}$$

Persamaan (1) menyatakan fungsi objektif minimasi makespan, sedangkan Persamaan (2) menyatakan fungsi objektif minimasi total tardiness. Persamaan (3) memastikan urutan operasi internal setiap job terpenuhi. Kendala (4) dan (5) memodelkan kapasitas mesin agar dua operasi pada mesin yang sama tidak diproses secara bersamaan. Persamaan (6) mendefinisikan batas atas *completion time* seluruh operasi untuk membentuk C_{max}. Persamaan (7) menghubungkan *completion time* job dengan operasi terakhirnya. Persamaan (8) mendefinisikan tardiness terhadap due date. Persamaan (9) menetapkan domain variabel kontinu nonnegatif dan variabel biner urutan.

METODE HEURISTIK

Algoritma 1 menunjukkan alur umum heuristik yang dipakai pada penelitian ini. Untuk objektif makespan, prosesnya dibuat ringkas: membangun jadwal awal berbasis EST lalu memperbaikinya melalui *adjacent swap*. Untuk objektif tardiness, alur diperluas dalam dua fase, yaitu fase diversifikasi dan fase intensifikasi. Fase diversifikasi membangkitkan banyak kandidat awal dari kombinasi parameter ATC dan variasi tie-breaking acak sehingga ruang pencarian yang dijelajahi lebih luas. Fase intensifikasi kemudian memfokuskan pencarian di sekitar solusi elite melalui perturbasi terarah pada job tardy dan pencarian lokal berulang sampai batas iterasi. Struktur dua fase ini bertujuan menyeimbangkan eksplorasi dan eksploitasi agar kualitas tardiness meningkat tanpa membuat

kerangka heuristik menjadi terlalu kompleks.

Algorithm 1 Heuristik umum JSSP untuk objektif makespan dan tardiness

Require: instance I, objektif obj, batas iterasi pencarian lokal N

if obj = makespan **then**
  Bangun satu jadwal awal dengan aturan dispatch *earliest start time* (EST)
  Lakukan pencarian lokal berbasis *adjacent swap*
  return jadwal terbaik
else
  Bangun himpunan solusi awal dengan grid parameter ATC
  Tambahkan variasi tie-breaking stokastik untuk diversifikasi
  for setiap solusi awal **do**
    Jalankan pencarian lokal cepat berbasis *insertion/swap*
  end for
  Pilih beberapa solusi elite terbaik
  for setiap solusi elite **do**
    for $r = 1$ sampai R **do**
      Lakukan perturbasi yang memprioritaskan job tardy
      Jalankan pencarian lokal untuk intensifikasi
      Perbarui solusi terbaik jika objektif membaik
    end for
  end for
  Jalankan pencarian lokal final sampai iterasi maksimum N
  return jadwal terbaik
end if

Pada tahap konstruksi jadwal awal, mekanisme dispatch disesuaikan terhadap objektif. Untuk objektif makespan, operasi yang dipilih pada setiap langkah adalah operasi feasible dengan EST terkecil, lalu tie-break menggunakan waktu proses yang lebih pendek agar utilisasi mesin tetap tinggi dan blocking antarmesin berkurang. Untuk objektif tardiness, prioritas tidak hanya ditentukan oleh waktu mulai paling awal, tetapi juga oleh risiko keterlambatan job melalui skor bergaya ATC pada Persamaan (10). Skor ini menggabungkan komponen slack, bobot risiko lateness, dan bias terhadap kandidat dengan EST yang lebih dekat, sehingga konstruksi awal lebih sensitif terhadap due date dibandingkan aturan berbasis waktu proses murni. Nilai parameter implementasi ditetapkan pada $k = 2.0$, $\beta = 0.15$, dan $\gamma = 0.01$.

$$\text{score}(o) = \frac{\exp\left(-\frac{\max(0, \text{slack}_j)}{k\bar{p}}\right)}{p_o} + \beta \frac{\max(0, -\text{slack}_j)}{\bar{p}} - \gamma \frac{EST_o - EST_{\min}}{\bar{p}}, \tag{10}$$

Setelah jadwal awal dibentuk, tahap perbaikan lokal diterapkan secara iteratif pada urutan operasi di setiap mesin. Untuk objektif makespan, neighborhood yang digunakan adalah adjacent swap karena biaya evaluasinya rendah dan efektif memperbaiki bottleneck lokal pada critical path. Untuk objektif tardiness, neighborhood diperluas menjadi kombinasi insertion move dan pairwise swap pada mesin yang sama agar perubahan urutan dapat lebih agresif terhadap job-job kritis due date. Setiap kandidat langkah dievaluasi ulang terhadap kelayakan precedence dan kapasitas mesin, lalu diterima hanya jika menghasilkan perbaikan objektif, sehingga proses pencarian bergerak secara monoton menuju solusi lokal yang lebih baik.

Untuk memperkuat eksplorasi pada objektif tardiness, pencarian dijalankan dari banyak titik awal yang dibentuk melalui grid parameter ATC dan variasi tie-breaking stokastik. Seluruh solusi awal diproses terlebih dahulu menggunakan pencarian lokal cepat, lalu dipilih beberapa solusi elite dengan nilai tardiness terbaik. Dari setiap elite, algoritma melakukan siklus perturbation + local search secara berulang, dengan perturbasi yang diarahkan terutama pada operasi milik job tardy agar peluang keluar dari local optimum meningkat. Mekanisme multi-start ini menjaga keseimbangan antara

keragaman solusi awal dan intensifikasi di sekitar kandidat terbaik, sehingga kualitas akhir lebih andal dibandingkan skema satu titik awal.

HASIL DAN PEMBAHASAN

Bagian ini menyajikan rancangan eksperimen, hasil komputasi, serta diskusi secara terpadu agar alur evaluasi dari setup hingga interpretasi temuan dapat diikuti secara ringkas. Eksperimen menggunakan 10 instance sintetis. Setiap instance terdiri atas 10 job, 5 mesin, dan 5 operasi per job, dengan waktu proses operasi berupa bilangan bulat pada rentang [1,20]. Desain ini dipilih untuk menjaga keseimbangan antara kompleksitas disjunktif yang cukup menantang dan waktu komputasi yang masih realistis untuk perbandingan berulang antara MILP dan heuristik.

Due date setiap job dibangkitkan dari kombinasi lower bound makespan global dan total beban kerja job agar tingkat keketatan due date tidak terlalu longgar maupun terlalu ketat. Formulasi due date job j dinyatakan pada Persamaan (11). Komponen $LB_{C_{max}}$, W_j, dan distribusi pengali acak u_j dinyatakan pada Persamaan (12).

$$d_j = \max\{1,\ \mathrm{round}[(0.65\, LB_{C_{\max}} + 0.8\, W_j)u_j]\}, \tag{11}$$

$$LB_{C_{\max}} = \frac{\sum_{o\in\mathcal{O}} p_o}{|\mathcal{M}|}, \qquad W_j = \sum_{o\in\mathcal{O}_j} p_o\,, \qquad u_j \sim \mathcal{U}[0.9,1.2]. \tag{12}$$

Konstruksi ini memastikan heterogenitas tingkat kesulitan tardiness antar instance sehingga evaluasi heuristik lebih informatif.

Penyelesaian eksak dilakukan menggunakan Gurobi pada formulasi MILP dengan batas waktu 20 detik per instance. Pada sisi pendekatan aproksimasi, heuristik diimplementasikan dalam Python dengan batas maksimum pencarian lokal 200 iterasi per instance. Untuk objektif makespan digunakan konstruksi cepat berbasis dispatch dan perbaikan lokal sederhana, sedangkan untuk objektif tardiness digunakan varian heuristik yang diperkuat melalui multi-start dan iterated local search. Konfigurasi ini sengaja dipilih untuk menilai trade-off kualitas solusi dan waktu komputasi pada dua objektif yang memiliki karakter berbeda.

Evaluasi dilakukan pada empat indikator utama, yaitu nilai objektif, runtime, selisih absolut objektif antara heuristik dan MILP, serta gap relatif terhadap MILP. Definisi selisih absolut pada instance s dinyatakan pada Persamaan (13). Definisi gap relatif pada instance dinyatakan pada Persamaan (14). Untuk pelaporan agregat antar instance, rata-rata *diff* dihitung pada seluruh instance, sedangkan rata-rata gap dihitung hanya pada instance dengan nilai objektif MILP positif agar tidak terjadi pembagian dengan nol. Catatan terminologi: istilah diff pada tabel dan narasi selanjutnya merupakan singkatan dari selisih absolut sebagaimana didefinisikan pada Persamaan (13). Rata-rata gap yang hanya dihitung pada instance dengan nilai MILP positif ini selanjutnya disebut conditional average gap atau rata-rata gap terdefinisi untuk membedakannya dari rata-rata gap penuh yang dapat bias akibat pembagian nol.

$$\mathrm{Diff}^{(s)} = \mathrm{Obj}^{(s)}_{\mathrm{Heur}} - \mathrm{Obj}^{(s)}_{\mathrm{MILP}}, \tag{13}$$

$$\mathrm{Gap}^{(s)}(\%) = \frac{\mathrm{Obj}^{(s)}_{\mathrm{Heur}} - \mathrm{Obj}^{(s)}_{\mathrm{MILP}}}{\mathrm{Obj}^{(s)}_{\mathrm{MILP}}} \times 100\%, \quad \text{untuk } \mathrm{Obj}^{(s)}_{\mathrm{MILP}} > 0. \tag{14}$$

Tabel 2 merangkum performa agregat untuk 10 instance. Secara agregat, Tabel 2 memperlihatkan bahwa heuristik untuk objektif makespan berada pada posisi yang pragmatis: kualitasnya cukup dekat dengan MILP (rata-rata gap 9.25% dan median 6.28%) dengan biaya komputasi yang sangat rendah (0.003 detik dibandingkan 0.719 detik untuk MILP). Kombinasi ini menunjukkan bahwa untuk indikator kinerja utama (KPI) berbasis throughput global, heuristik dapat berfungsi sebagai baseline operasional yang kuat karena mempertahankan kualitas dalam rentang yang masih terkendali sambil memberi percepatan komputasi yang sangat besar.

Tabel 2 Ringkasan agregat hasil benchmark

Objektif	Rata-rata gap (%)	Median gap (%)	Rata-rata diff	Rata-rata runtime MILP (s)	Rata-rata runtime heuristik (s)
Makespan	9.25	6.28	11.9	0.719	0.003
Tardiness	52.39	18.83	2.4	1.664	9.96

Untuk objektif tardiness pada mode balanced, tabel yang sama menunjukkan trade-off yang lebih moderat. Rata-rata gap terdefinisi berada pada 52.39% dengan median 18.83% dan rata-rata diff 2.40, sementara runtime heuristik 9.960 detik masih lebih tinggi dibandingkan MILP 1.664 detik. Kesenjangan antara rata-rata dan median gap menunjukkan distribusi galat yang tidak simetris, yaitu sebagian instance relatif dekat dengan MILP tetapi beberapa instance masih memicu deviasi besar.

Interpretasi rinci pada Tabel 3 mengonfirmasi bahwa performa heuristik makespan cukup konsisten: tujuh dari sepuluh instance memiliki gap di bawah atau sama dengan sekitar 10%, dan salah satu instance mencapai solusi identik dengan MILP (gap 0%). Penyimpangan terbesar terkonsentrasi pada salah satu instance dengan gap 34.11% (diff 44), yang berperan sebagai pencilan dominan terhadap rata-rata. Dari sisi waktu komputasi, seluruh baris menunjukkan pola yang stabil, yaitu heuristik berada pada orde 0.00–0.01 detik, sedangkan MILP berada pada orde 0.15–1.52 detik. Temuan ini mempertegas bahwa keunggulan utama heuristik makespan terletak pada kecepatan yang sangat tinggi dengan risiko kualitas yang terutama muncul pada instance tertentu. Deviasi besar pada instance pencilan tersebut (gap 34.11%, diff 44) kemungkinan disebabkan oleh struktur urutan operasi yang menciptakan bottleneck mesin padat: ketika beberapa job bersaing pada mesin yang sama dalam jendela waktu yang sempit, heuristik berbasis EST cenderung memilih urutan yang secara lokal optimal namun menciptakan konflik penjadwalan kumulatif sehingga makespan membesar secara tidak proporsional. Karakteristik ini mengindikasikan bahwa penambahan move type yang lebih agresif, misalnya block move pada critical path, dapat secara spesifik menargetkan dan memperbaiki instance dengan profil bottleneck tinggi.

Tabel 3 Perbandingan MILP vs heuristik untuk objektif makespan

Instance	MILP Cmax	Heur Cmax	Diff	Gap (%)	Runtime MILP	Runtime Heur	Status MILP
1	150	159	9	6	0.72	0	OPTIMAL
2	145	151	6	4.14	1.52	0	OPTIMAL
3	121	127	6	4.96	0.45	0	OPTIMAL
4	114	130	16	14.04	0.99	0	OPTIMAL
5	130	139	9	6.92	0.37	0	OPTIMAL
6	137	146	9	6.57	0.85	0.01	OPTIMAL
7	127	140	13	10.24	0.59	0.01	OPTIMAL
8	129	173	44	34.11	0.15	0	OPTIMAL
9	126	133	7	5.56	0.93	0.01	OPTIMAL
10	137	137	0	0	0.6	0	OPTIMAL

Tabel 4 menunjukkan bahwa perilaku heuristik tardiness pada konfigurasi eksperimen ini masih heterogen. Pada empat instance (6, 7, 8, dan 10), heuristik mencapai nilai tardiness yang sama dengan MILP (diff 0), sedangkan pada instance 2, 5, dan 9 deviasinya masih kecil (diff 1–2). Tiga instance menjadi sumber gap besar, yaitu instance 1 (33.33%, diff 4), instance 3 (200%, diff 2), dan instance 4 (144.44%, diff 13). Untuk instance 3, nilai persentase tampak sangat tinggi karena nilai objektif MILP sangat kecil (total tardiness = 1), sehingga kenaikan absolut kecil langsung memperbesar gap relatif. Kolom runtime menunjukkan bahwa heuristik tardiness berjalan pada kisaran 8.27–13.28 detik dan pada mayoritas instance masih lebih lambat daripada MILP. Kegagalan relatif pada instance 1 (gap 33.33%) dan instance 4 (gap 144.44%, diff 13) kemungkinan dipicu oleh machine bottleneck congestion: beberapa job dengan due date berdekatan bergantung pada mesin yang sama secara bersamaan, sehingga skor ATC tidak cukup membedakan urutan optimal dan tardiness terakumulasi pada job-job yang paling sensitif terhadap keterlambatan. Pada instance 4, ruang solusi yang sempit akibat ketatnya due date membuat multi-start belum mampu menemukan solusi mendekati MILP dalam batas iterasi yang ditetapkan. Pada instance 3, meskipun gap persentase sangat besar (200%), deviasi absolut hanya 2 unit, menunjukkan kedekatan absolut yang baik namun sensitivitas persentase tinggi akibat denominatornya yang sangat kecil (total tardiness MILP = 1). Ketiga temuan ini mengindikasikan bahwa augmentasi dengan block move atau ejection chain dapat secara khusus mengatasi skenario bottleneck padat dan memperbaiki robustitas heuristik

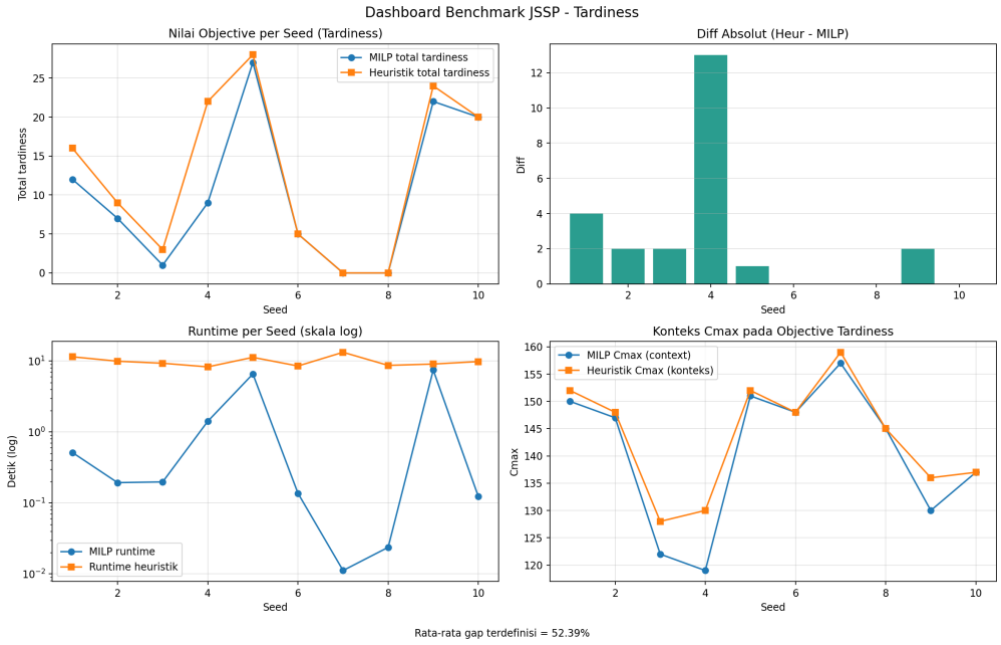
tardiness pada instance-instance tersebut.

Tabel 4 Perbandingan MILP vs heuristik untuk objektif tardiness

Instance	MILP Tardiness	Heur Tardiness	Diff	Gap (%)	MILP Cmax	Heur Cmax	Runtime MILP	Runtime Heur	Status MILP
1	12	16	4	33.33	150	152	0.51	11.47	OPTIMAL
2	7	9	2	28.57	147	148	0.19	9.93	OPTIMAL
3	1	3	2	200	122	128	0.2	9.3	OPTIMAL
4	9	22	13	144.44	119	130	1.41	8.27	OPTIMAL
5	27	28	1	3.7	151	152	6.56	11.27	OPTIMAL
6	5	5	0	0	148	148	0.14	8.51	OPTIMAL
7	0	0	0		157	159	0.01	13.28	OPTIMAL
8	0	0	0		145	145	0.02	8.68	OPTIMAL
9	22	24	2	9.09	130	136	7.46	9.07	OPTIMAL
10	20	20	0	0	137	137	0.12	9.82	OPTIMAL

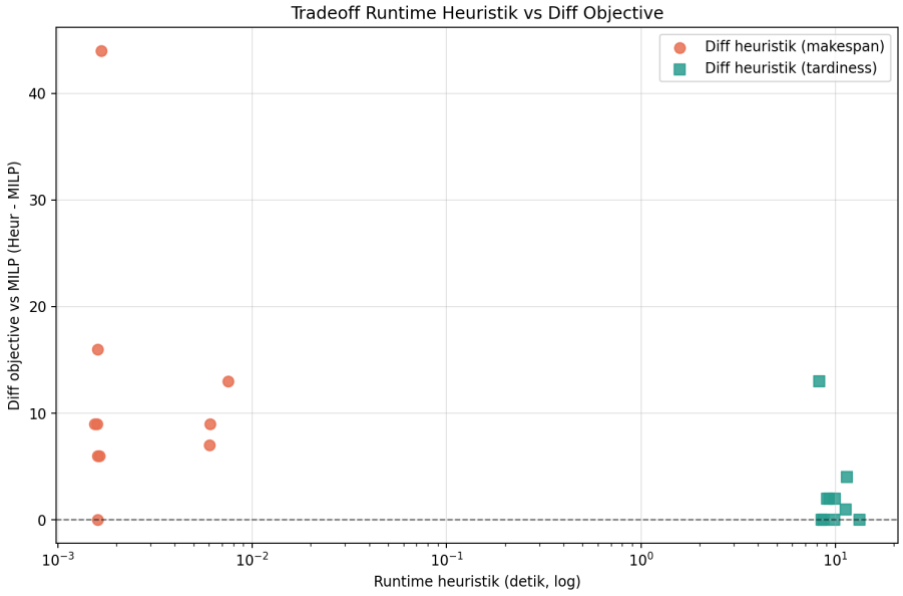
Gambar 1 menunjukkan bahwa untuk objektif makespan, performa heuristik relatif stabil pada sebagian besar instance dengan gap moderat, meskipun tetap muncul pencilan yang jelas pada instance 8. Pola ini konsisten dengan ringkasan agregat: heuristik memberikan kecepatan yang sangat tinggi, tetapi kualitas solusi dapat menurun tajam pada instance tertentu. Pada sisi tardiness (Gambar 2), pola yang tampak lebih berfluktuasi: beberapa instance mencapai hasil sangat dekat dengan MILP (misalnya gap 0%), sementara instance lain tetap menghasilkan gap besar (misalnya instance 3 dan instance 4). Nilai agregat saat ini menempatkan rata-rata gap terdefinisi pada 52.39% dengan robustitas antarinstance yang masih belum merata.

Gambar 1 Dashboard kinerja untuk objektif makespan.

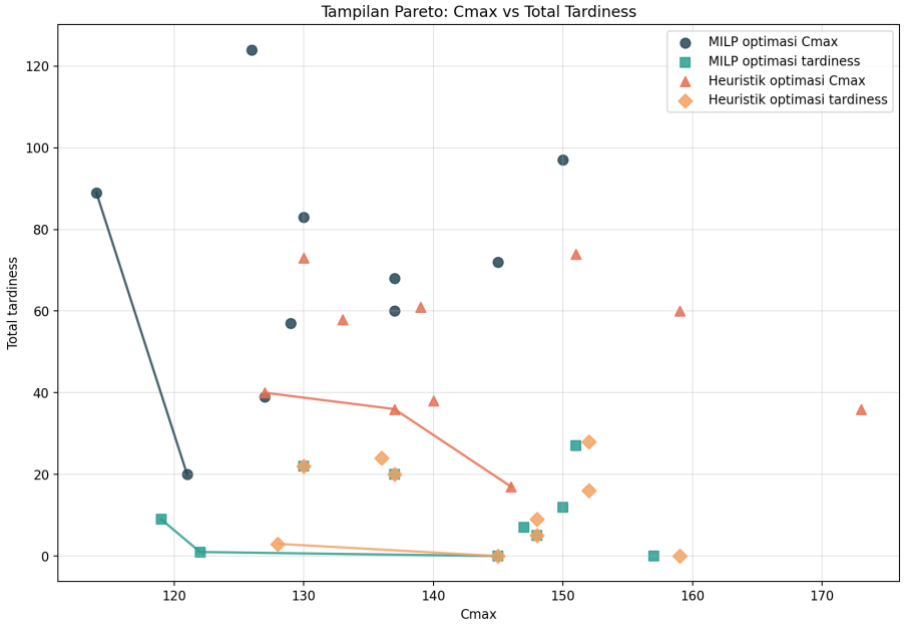


Gambar 2 Dashboard kinerja untuk objektif tardiness.

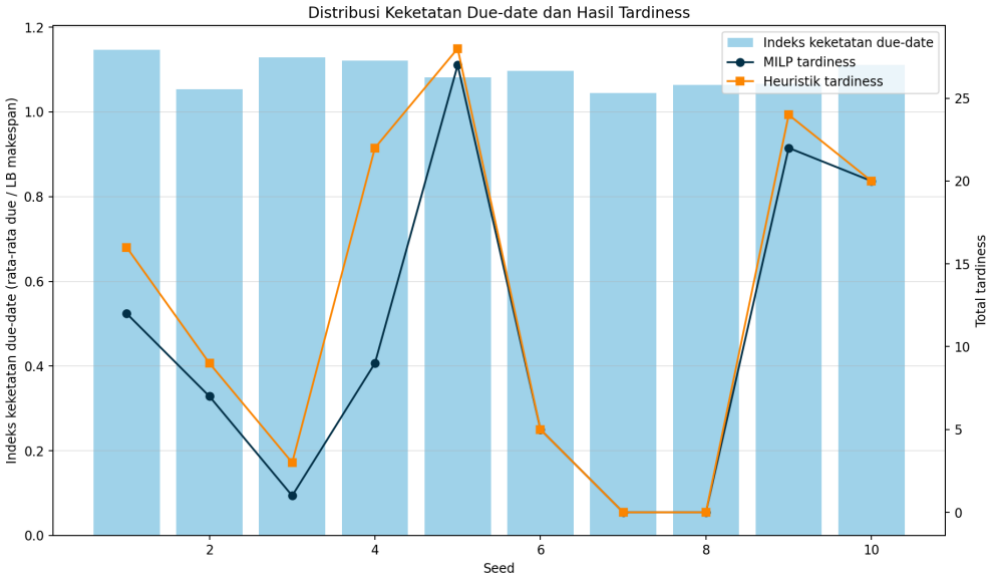
Gambar 3 memperjelas konsekuensi komputasi pada objektif tardiness: runtime heuristik berada pada level menengah-tinggi dan kualitas masih berfluktuasi antar instance sehingga belum sepenuhnya konsisten mendekati MILP. Selanjutnya, Gambar 4 menegaskan adanya trade-off klasik antara C_max dan total tardiness, sehingga solusi yang baik pada satu objektif belum tentu dominan pada objektif lain. Terakhir, Gambar 5 memberi indikasi bahwa keketatan due date berperan terhadap sensitivitas tardiness; instance dengan due date yang lebih ketat cenderung lebih sulit ditangani secara konsisten oleh heuristik.



Gambar 3 Trade-off runtime heuristik terhadap diff objektif.

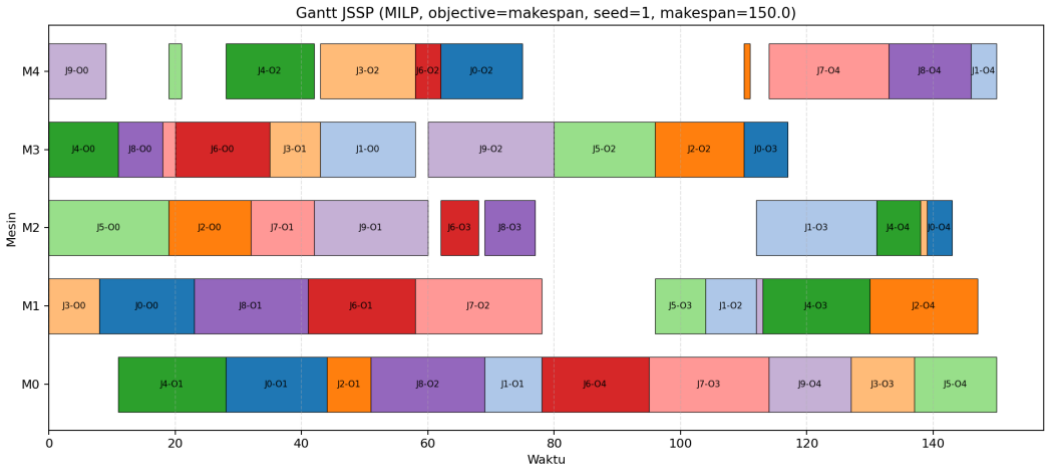


Gambar 4 Tampilan Pareto antara C_{max} dan total tardiness

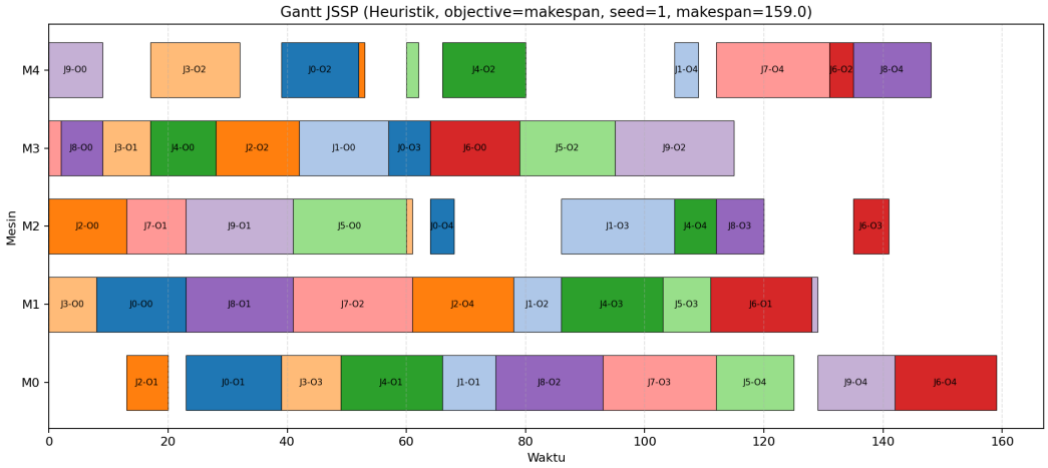


Gambar 5 Distribusi keketatan due date dan capaian tardiness.

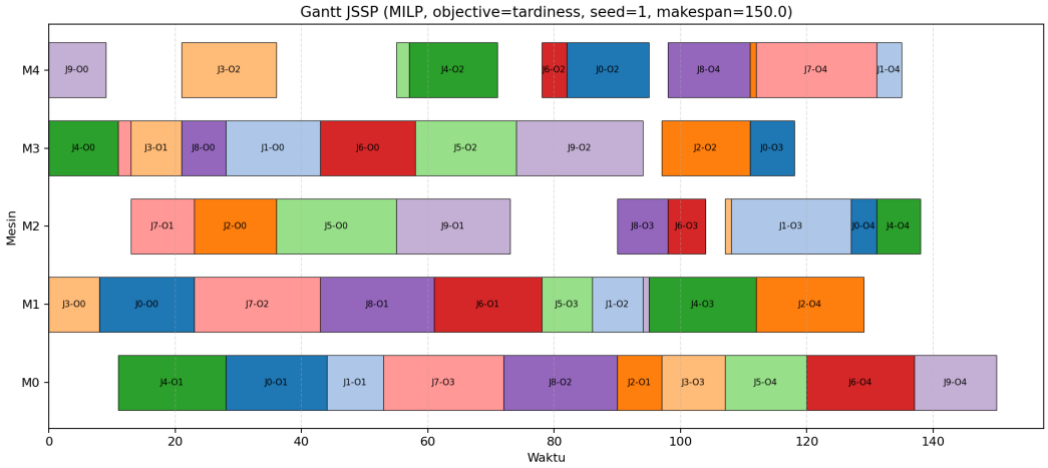
Untuk melengkapi pembacaan metrik numerik, Gambar 6 sampai Gambar 9 menampilkan struktur jadwal pada instance 1 untuk masing-masing objektif, dengan urutan MILP terlebih dahulu lalu heuristik. Visualisasi ini membantu melihat pola utilisasi mesin, urutan operasi antarmesin, dan tingkat fragmentasi blok proses yang tidak langsung terlihat dari tabel agregat.



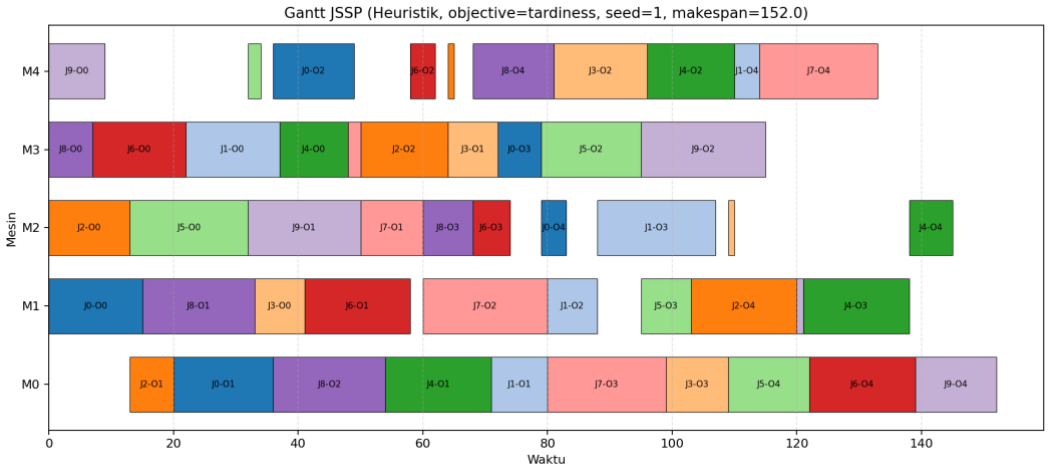
Gambar 6 Gantt chart MILP untuk objektif makespan (instance 1).



Gambar 7 Gantt chart heuristik untuk objektif makespan (instance 1).



Gambar 8 Gantt chart MILP untuk objektif tardiness (instance 1).



Gambar 9 Gantt chart heuristik untuk objektif tardiness (instance 1).

Hasil eksperimen memperlihatkan bahwa karakter objektif sangat menentukan perilaku algoritme. Pada objektif makespan, lanskap solusi cenderung lebih ramah terhadap aturan dispatch sederhana, sehingga heuristik dapat mempertahankan kualitas yang cukup dekat dengan MILP dengan biaya komputasi yang sangat kecil. Hal ini konsisten dengan temuan per- instance yang relatif stabil, kecuali beberapa instance pencilan. Sebaliknya, pada objektif tardiness, perubahan urutan lokal dapat menimbulkan dampak yang jauh lebih nonlinier terhadap keterlambatan total karena interaksi antara due date, urutan operasi, dan bottleneck mesin menjadi lebih sensitif. Akibatnya, variabilitas antar instance tetap tinggi walaupun beberapa instance mencapai solusi identik dengan MILP.

Dari sisi metodologis, konfigurasi yang digunakan menunjukkan kompromi antara kualitas dan waktu komputasi pada objektif tardiness. Gap terhadap MILP masih muncul pada sebagian instance, terutama ketika struktur due date dan bottleneck mesin menghasilkan lanskap solusi yang tajam. Artinya, perbaikan kualitas tardiness tetap membutuhkan biaya komputasi, dan intensitas pencarian perlu dipilih secara selektif sesuai karakter instance. Temuan ini menegaskan pentingnya desain heuristik yang adaptif, misalnya dengan intensifikasi dinamis hanya pada instance dengan indikasi risiko tardiness tinggi, agar rasio kualitas-waktu lebih efisien.

Implikasi praktis dari hasil ini adalah perlunya kebijakan pemilihan solver berbasis tujuan operasional. Jika target utama adalah throughput dan respons cepat untuk penjadwalan harian, heuristik makespan dapat diposisikan sebagai baseline yang kuat. Namun, jika KPI utama adalah keterlambatan terhadap due date, terutama pada lingkungan dengan penalti keterlambatan tinggi, pendekatan eksak atau hybrid tetap lebih relevan. Dalam konteks ini, heuristik tardiness lebih tepat dilihat sebagai mesin pembangkit solusi awal berkualitas untuk kemudian dipoles oleh MILP, bukan sebagai pengganti penuh solver eksak. Secara konkret, implementasi heuristik makespan pada lingkungan produksi manufaktur berpotensi memangkas waktu penjadwalan harian dari hitungan menit (solver MILP) menjadi hitungan milidetik, sehingga revisi jadwal secara real-time saat terjadi gangguan mesin (machine breakdown) menjadi layak secara operasional. Dari sisi efisiensi biaya, pengurangan makespan yang konsisten berkontribusi pada peningkatan utilisasi mesin secara keseluruhan, yang dalam konteks produksi kontinu dapat diterjemahkan sebagai penurunan biaya overhead per unit produk. Dari perspektif integrasi sistem, heuristik berbasis aturan dispatch memiliki keunggulan implementabilitas lebih tinggi dibandingkan solver MILP komersial dalam arsitektur sistem ERP seperti SAP Production Planning atau Oracle Manufacturing Cloud, karena dapat diimplementasikan sebagai modul penjadwalan ringan tanpa lisensi solver tambahan dan dengan latensi eksekusi yang kompatibel dengan siklus perencanaan harian.

KESIMPULAN

Penelitian ini menghasilkan baseline JSSP yang reprodusibel untuk dua objektif utama dan menunjukkan bahwa performa algoritme sangat bergantung pada definisi objektif. MILP tetap menjadi acuan kualitas terbaik secara konsisten. Heuristik yang diusulkan sangat efektif untuk makespan karena mampu memberikan solusi yang cukup baik dengan runtime sangat kecil, sedangkan pada

tardiness konfigurasi balanced memberikan kompromi praktis antara kualitas dan waktu komputasi meskipun belum menyamai kualitas MILP secara konsisten. Secara keseluruhan, kontribusi utama studi ini adalah penyediaan kerangka komparatif yang transparan, terukur, dan mudah direplikasi untuk mengevaluasi strategi eksak versus heuristik.

Keterbatasan utama studi ini terletak pada ukuran instance yang masih moderat, penggunaan data sintetis, dan skema tuning parameter yang belum adaptif terhadap karakter tiap instance. Arah pengembangan berikutnya difokuskan pada adaptive parameter control, perluasan neighborhood (misalnya block move atau ejection chain), serta integrasi matheuristic yang lebih erat antara heuristik dan MILP melalui *warm start* serta polishing terarah. Validasi pada data *shopfloor* riil juga penting agar temuan ini dapat ditransfer ke konteks industri dengan struktur due date dan variabilitas proses yang lebih kompleks.

DAFTAR PUSTAKA

Fatemi-Anaraki, S., Tavakkoli-Moghaddam, R., Foumani, M., & Vahedi-Nouri, B. (2023). Scheduling of Multi-Robot Job Shop Systems in Dynamic Environments: Mixed-Integer Linear Programming and Constraint Programming Approaches. *Omega*, *115*, 102770. https://doi.org/10.1016/j.omega.2022.102770

Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, *1*(2), 117–129. https://doi.org/10.1287/moor.1.2.117

Gui, Y., Tang, D., Zhu, H., Zhang, Y., & Zhang, Z. (2023). Dynamic scheduling for flexible job shop using a deep reinforcement learning approach. *Computers & Industrial Engineering*, *180*, 109255. https://doi.org/10.1016/j.cie.2023.109255

Jain, A. S., & Meeran, S. (1999). Deterministic job-shop scheduling: Past, present and future. *European Journal of Operational Research*, *113*(2), 390–434. https://doi.org/10.1016/S0377-2217(98)00113-1

Ku, W.-Y., & Beck, J. C. (2016). Mixed Integer Programming models for job shop scheduling: A computational analysis. *Computers & Operations Research*, *73*, 165–173. https://doi.org/10.1016/j.cor.2016.04.006

Laborie, P., Rogerie, J., Shaw, P., & Vilím, P. (2018). IBM ILOG CP optimizer for scheduling. *Constraints*, *23*(2), 210–250. https://doi.org/10.1007/s10601-018-9281-x

Lv, L., Zhang, C., Fan, J., & Shen, W. (2025). Deep reinforcement learning for job shop scheduling problems: A comprehensive literature review. *Knowledge-Based Systems*, *321*, 113633. https://doi.org/10.1016/j.knosys.2025.113633

Manne, A. S. (1960). On the Job-Shop Scheduling Problem. *Operations Research*, *8*(2), 219–223. https://doi.org/10.1287/opre.8.2.219

Pinedo, M. L. (2016). Job Shops (Deterministic). In M. L. Pinedo (Ed.), *Scheduling: Theory, Algorithms, and Systems* (pp. 183–220). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-26580-3_7

Vepsalainen, A. P. J., & Morton, T. E. (1987). Priority Rules for Job Shops with Weighted Tardiness Costs. *Management Science*, *33*(8), 1035–1047. https://doi.org/10.1287/mnsc.33.8.1035

Wang, L., Hu, X., Wang, Y., Xu, S., Ma, S., Yang, K., … Wang, W. (2021). Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning. *Computer Networks*, *190*, 107969. https://doi.org/10.1016/j.comnet.2021.107969

Xiong, H., Shi, S., Ren, D., & Hu, J. (2022). A survey of job shop scheduling problem: The types and models. *Computers & Operations Research*, *142*, 105731. https://doi.org/10.1016/j.cor.2022.105731