



Penyetelan Parameter Berbantuan Machine Learning untuk Penjadwalan Cloud Berbasis Metaheuristik

Received : April 05, 2026; Revised : April 23, 2026; Accepted : Jul 07, 2026; Published : Sep 15, 2026

Ardi Pujiyanta^{1*}, Faisal Fajri Rahani², Taufiq Ismail³

¹Magister Teknik Elektro, Universitas Ahmad Dahlan Yogyakarta,
Jl. Ki Ageng Pemanahan, Kragilan, Tamanan, Kec. Banguntapan, Kabupaten Bantul, Yogyakarta, Indonesia
ardipujiyanta@tif.uad.ac.id

²Informatika, Universitas Ahmad Dahlan Yogyakarta,
Jl. Ki Ageng Pemanahan, Kragilan, Tamanan, Kec. Banguntapan, Kabupaten Bantul, Yogyakarta, Indonesia
faisal.fajri@tif.uad.ac.id

³Informatika, Universitas Ahmad Dahlan Yogyakarta,
Jl. Ki Ageng Pemanahan, Kragilan, Tamanan, Kec. Banguntapan, Kabupaten Bantul, Yogyakarta, Indonesia
taufiq@tif.uad.ac.id

*Corresponding Author

Abstrak:

Penjadwalan *cloud* berbasis metaheuristik sangat dipengaruhi oleh kualitas pemilihan parameter algoritma. Namun, pada banyak penelitian, parameter *genetic algorithm* masih ditetapkan secara tetap atau dituning secara konvensional, sehingga kurang adaptif terhadap perubahan karakteristik *workload*. Penelitian ini mengusulkan pendekatan penyetelan parameter berbantuan *machine learning* untuk penjadwalan *cloud* berbasis metaheuristik guna memprediksi parameter *genetic algorithm* berdasarkan karakteristik *workload cloud*. Dataset yang digunakan berasal dari GoCJ mentah yang ditransformasikan menjadi 45 skenario *workload* melalui tiga *regime*, yaitu *tight*, *medium*, dan *loose*. Tahapan penelitian meliputi pembentukan fitur *workload*, pelabelan parameter terbaik *genetic algorithm*, pembangunan model prediksi parameter, serta evaluasi dampaknya terhadap performa penjadwalan. Metode usulan, yaitu GA-ML, dibandingkan dengan FCFS, EDF, GA-Default, dan GA-Grid. Hasil eksperimen menunjukkan bahwa GA-ML memberikan performa lebih baik dibandingkan GA-Grid pada tiga dari empat metrik evaluasi. Secara khusus, GA-ML menurunkan *total penalty* sebesar 8,269%, *weighted tardiness* sebesar 8,804%, dan *violation rate* sebesar 1,366% dibandingkan GA-Grid. Namun, GA-ML menghasilkan *makespan* sedikit lebih tinggi sebesar 0,582%. Hasil uji Wilcoxon *signed-rank* menunjukkan bahwa perbedaan antara GA-ML dan GA-Grid signifikan pada seluruh metrik. Temuan ini menunjukkan bahwa penyetelan parameter berbantuan *machine learning* efektif untuk meningkatkan performa penjadwalan berbasis SLA, terutama dalam menekan penalti, keterlambatan berbobot, dan tingkat pelanggaran *deadline*, meskipun terdapat *trade-off* kecil pada *makespan*.

Kata Kunci:

Penjadwalan Cloud, Machine Learning, Genetic Algorithm, Penyetelan Parameter, Metaheuristik.

Abstract:

Metaheuristic-based cloud scheduling is strongly influenced by the quality of algorithm parameter selection. However, in many studies, genetic algorithm parameters are still fixed manually or tuned using conventional strategies, making them less adaptive to changes in workload characteristics. This study proposes a machine learning-assisted parameter tuning approach for metaheuristic-based cloud scheduling to predict genetic algorithm parameters based on cloud workload characteristics. The dataset was derived from raw GoCJ data and transformed into 45 workload scenarios under three regimes: tight, medium, and loose. The research stages include workload feature construction, best-parameter labeling for the genetic algorithm, parameter prediction model development, and evaluation of its impact on scheduling performance. The proposed method, namely GA-ML, was compared with FCFS, EDF, GA-Default, and GA-Grid. The experimental results show that GA-ML



outperforms GA-Grid in three out of four evaluation metrics. Specifically, GA-ML reduces total penalty by 8.269%, weighted tardiness by 8.804%, and violation rate by 1.366% compared with GA-Grid. However, GA-ML produces a slightly higher makespan, with a degradation of 0.582%. The Wilcoxon signed-rank test confirms that the differences between GA-ML and GA-Grid are statistically significant across all evaluated metrics. These findings indicate that machine learning-assisted parameter tuning is effective in improving SLA-oriented scheduling performance, particularly in reducing penalty, weighted tardiness, and deadline violation rate, although it introduces a small trade-off in makespan.

Keywords:

Cloud Scheduling, Machine Learning, Genetic Algorithm, Parameter Tuning, Metaheuristics.

1. Pendahuluan

Cloud computing telah menjadi infrastruktur penting untuk menyediakan layanan komputasi yang elastis, terukur, dan tersedia sesuai permintaan. Dalam praktiknya, kualitas layanan cloud sangat dipengaruhi oleh kemampuan sistem dalam menjadwalkan *job* dan mengalokasikan sumber daya secara efisien di tengah *workload* yang heterogen, dinamis, dan sering kali berubah antar-skenario [1]–[5]. Oleh karena itu, *cloud scheduling* tidak lagi dipahami hanya sebagai proses penentuan urutan eksekusi, tetapi sebagai mekanisme pengambilan keputusan yang harus menyeimbangkan efisiensi waktu penyelesaian, utilisasi *resource*, dan biaya operasional akibat keterlambatan [3], [6]–[8].

1.1. Perkembangan Penjadwalan dan Optimasi pada Lingkungan Cloud

Penelitian tentang penjadwalan dan alokasi sumber daya pada lingkungan *cloud* terus berkembang seiring meningkatnya kompleksitas sistem komputasi modern. Tinjauan literatur terbaru menunjukkan bahwa fokus penelitian telah bergeser dari heuristik sederhana menuju pendekatan yang lebih adaptif, cerdas, dan berbasis optimasi [1]–[3]. Dalam konteks ini, sejumlah studi menyoroti pentingnya *resource scheduling* yang mampu menyesuaikan diri terhadap variasi *workload*, *multi-scenario resource management*, serta kondisi pusat data yang semakin heterogen [6]–[9].

Di sisi lain, algoritma metaheuristik tetap menjadi salah satu pendekatan yang paling banyak digunakan dalam *cloud scheduling* karena mampu menangani ruang solusi yang besar dan kompleks. Berbagai penelitian telah mengusulkan metode berbasis *frilled lizard optimization*, *honey badger optimization*, *particle swarm optimization*, *probabilistic genetic algorithm*, *whale optimization*, *hybrid GA-GWO*, dan *improved multi-verse optimizer* untuk meningkatkan kualitas *task scheduling* maupun *resource scheduling* [10]–[16]. Studi-studi tersebut menunjukkan bahwa metaheuristik masih relevan, terutama untuk masalah penjadwalan multi-objektif dan lingkungan cloud yang bersifat heterogen [15], [17]–[19].

Meskipun demikian, keberhasilan pendekatan metaheuristik tidak hanya ditentukan oleh jenis algoritma yang digunakan, tetapi juga oleh kualitas konfigurasi parameternya. Pada genetic algorithm dan algoritma sejenis, parameter seperti crossover rate, mutation rate, elitism, dan tournament size sangat memengaruhi keseimbangan antara eksplorasi dan eksploitasi dalam proses pencarian solusi [13], [15], [18]. Permasalahannya, banyak penelitian *cloud scheduling* lebih menitikberatkan pada usulan algoritma baru, sedangkan persoalan tuning parameter sering masih diperlakukan sebagai pengaturan tetap atau ditentukan secara manual [15], [17], [18], [20].

1.2. Peran Machine Learning dalam Cloud Scheduling

Sejalan dengan perkembangan tersebut, *machine learning* semakin banyak dimanfaatkan untuk mendukung pengambilan keputusan pada manajemen *cloud*. Beberapa penelitian mengkaji *resource allocation* berbasis *machine learning*, *AI-driven job scheduling*, prediksi kondisi sistem, serta optimasi keputusan eksekusi agar lebih adaptif terhadap karakteristik *workload* [21]–[25]. Pendekatan ini menjadi penting karena lingkungan cloud modern menghasilkan data operasional yang kaya dan dapat

digunakan untuk membangun model prediktif yang membantu sistem mengambil keputusan lebih baik [1], [23], [26].

Selain itu, integrasi *machine learning* dengan mekanisme optimasi juga mulai banyak dikembangkan. Sejumlah studi menunjukkan bahwa pembelajaran dapat dipadukan dengan reinforcement learning, prediksi *resource*, maupun *hybrid learning* untuk meningkatkan kualitas penjadwalan pada *cloud* dan *edge-cloud* [9], [22], [26]–[29]. Temuan ini memperlihatkan bahwa *machine learning* tidak harus selalu menggantikan *scheduler* secara penuh, tetapi juga dapat ditempatkan sebagai lapisan pendukung keputusan yang membantu sistem memahami pola *workload* dan memilih tindakan yang lebih sesuai [21], [23], [25], [29].

Namun demikian, sebagian besar penelitian *machine learning* pada *cloud* masih berfokus pada *resource allocation*, *scheduling policy* langsung, atau optimasi keputusan eksekusi secara end-to-end [22], [23], [25], [26], [28]. Pendekatan tersebut memang penting, tetapi belum banyak yang secara eksplisit memanfaatkan *machine learning* untuk mempelajari konfigurasi parameter algoritma metaheuristik pada masalah *cloud scheduling* [30], [31]. Padahal, dari sudut pandang implementasi, pendekatan seperti ini berpotensi lebih praktis karena tetap mempertahankan struktur *scheduler* yang ada, sambil meningkatkan kualitas tuning parameternya secara otomatis.

1.3. Gap Penelitian dan Posisi Studi Ini

Berdasarkan tinjauan di atas, dapat dilihat bahwa terdapat dua arus besar dalam literatur *cloud scheduling*. Arus pertama berfokus pada pengembangan algoritma metaheuristik baru atau varian hibrida untuk memperbaiki performa penjadwalan [10], [11], [13], [15], [17], [18]. Arus kedua berfokus pada pemanfaatan *machine learning* untuk mendukung pengambilan keputusan *resource allocation* dan scheduling secara langsung [21]–[23], [25], [29]. Meskipun keduanya berkembang pesat, irisan di antara keduanya, yaitu penggunaan *machine learning* untuk membantu *tuning* parameter metaheuristik pada *cloud scheduling*, masih relatif terbatas [30], [31].

Kesenjangan ini menjadi penting karena parameter metaheuristik yang baik pada satu skenario *workload* belum tentu tetap efektif pada skenario lain. Variasi jumlah *job*, distribusi ukuran *job*, *tightness workload*, dan kondisi *resource* dapat menyebabkan konfigurasi parameter yang optimal berubah antar-kondisi [6]–[9]. Dengan demikian, penggunaan parameter tetap berisiko menghasilkan performa yang kurang konsisten, terutama ketika sistem dihadapkan pada *workload cloud* yang dinamis dan heterogen [1]–[4].

Berdasarkan celah tersebut, penelitian ini mengusulkan pendekatan penyetelan parameter berbantuan *machine learning* untuk penjadwalan cloud berbasis metaheuristik. Penelitian ini memposisikan *machine learning* bukan sebagai pengganti *scheduler* secara penuh, tetapi sebagai mekanisme prediktif untuk memetakan karakteristik *workload* ke konfigurasi parameter metaheuristik yang lebih sesuai. Dengan pendekatan ini, *scheduler* metaheuristik tetap dipertahankan sebagai inti mekanisme optimasi, sedangkan *machine learning* berperan sebagai lapisan pendukung keputusan untuk mengotomatisasi proses *tuning* parameter [25], [30], [31].

Secara khusus, penelitian ini diarahkan untuk menjawab apakah *tuning* parameter berbantuan *machine learning* mampu menghasilkan performa penjadwalan yang lebih kompetitif dibandingkan penggunaan parameter default maupun *tuning* konvensional. Dengan demikian, kontribusi utama penelitian ini terletak pada formulasi *tuning* parameter metaheuristik sebagai masalah prediksi, integrasi karakterisasi *workload* dengan proses *tuning*, serta evaluasi empiris terhadap dampaknya pada performa *cloud scheduling*. Oleh karena itu, bagian berikutnya menjelaskan metode penelitian yang digunakan, mulai dari pembentukan data eksperimen hingga prosedur evaluasi kinerja metode usulan.

2. Metode

Penelitian ini menggunakan pendekatan prediktif-komparatif untuk mengevaluasi tuning parameter berbantuan *machine learning* pada penjadwalan cloud berbasis metaheuristik. Berbeda dari penelitian yang menempatkan *machine learning* sebagai *scheduler end-to-end*, studi ini memposisikan *machine learning* sebagai lapisan pendukung keputusan untuk memprediksi konfigurasi parameter algoritma metaheuristik berdasarkan karakteristik *workload*. Pendekatan ini sejalan dengan perkembangan penelitian terbaru pada *cloud resource management*, *AI-driven scheduling*, dan integrasi *machine learning* untuk optimasi keputusan eksekusi di lingkungan cloud [1]–[3], [21], [23], [25]. Selain itu, literatur mengenai *machine learning* untuk meningkatkan metaheuristik juga menunjukkan bahwa pembelajaran dapat digunakan untuk membantu menentukan konfigurasi optimasi yang lebih efektif pada ruang pencarian yang kompleks [30], [31].

2.1 Desain Penelitian

Secara umum, alur penelitian terdiri atas enam tahap utama. Pertama, dataset mentah GoCJ dibaca dan dipilih menjadi sejumlah himpunan *workload* representatif. Kedua, setiap himpunan *workload* dibentuk menjadi beberapa skenario penjadwalan dengan tingkat keketatan yang berbeda. Ketiga, untuk setiap skenario dilakukan pencarian parameter *genetic algorithm* yang memberikan nilai objektif terbaik, sehingga diperoleh label parameter optimal. Keempat, dibangun model *machine learning* untuk memprediksi parameter tersebut dari fitur *workload*. Kelima, parameter hasil prediksi digunakan untuk menjalankan *scheduler* metaheuristik pada data uji. Keenam, kinerja metode usulan dibandingkan dengan beberapa metode pembandingan, yaitu FCFS, EDF, GA-Default, dan GA-Grid. Desain ini dipilih karena permasalahan utama penelitian bukan pada perancangan algoritma penjadwalan baru, melainkan pada otomatisasi tuning parameter untuk meningkatkan kualitas keputusan metaheuristik. Dengan demikian, penelitian ini memadukan karakterisasi *workload*, pembelajaran prediktif, dan evaluasi dampak scheduling dalam satu kerangka eksperimen yang konsisten [21], [25], [30], [31].

2.2 Dataset dan Pembentukan Skenario Workload

Dataset yang digunakan adalah dataset mentah GoCJ yang berisi ukuran *job* pada beberapa berkas *workload*. Pada penelitian ini digunakan 15 berkas *workload* mentah yang kemudian ditransformasikan menjadi 45 skenario, yaitu masing-masing *workload* dibentuk ke dalam tiga regime, yaitu tight, medium, dan loose. Strategi ini digunakan agar evaluasi tidak hanya bergantung pada satu kondisi beban kerja, tetapi juga menguji perilaku metode pada beberapa tingkat keketatan skenario. Setiap *job* direpresentasikan sebagai unit kerja yang memiliki ukuran komputasi, waktu kedatangan, deadline sintetis, dan bobot penalti. Karena dataset mentah GoCJ tidak secara langsung menyediakan atribut deadline dan penalti, kedua atribut tersebut dibentuk secara eksperimental agar penjadwalan dapat dievaluasi pada konteks yang lebih operasional, khususnya untuk metrik keterlambatan dan penalti. Pembentukan skenario seperti ini masih relevan dengan arah penelitian *cloud scheduling* modern yang menekankan pentingnya evaluasi pada *workload* heterogen, multi-skenario *resource management*, dan adaptive scheduling [1], [6], [7], [9].

Pada sisi infrastruktur, penelitian ini menggunakan sekumpulan mesin virtual heterogen yang direpresentasikan melalui kapasitas MIPS berbeda. Dengan setting ini, setiap skenario tidak hanya merefleksikan variasi *workload*, tetapi juga diuji pada lingkungan *resource* yang heterogen sebagaimana umum ditemukan pada *cloud scheduling* [1], [3].

2.3 Representasi Fitur dan Pelabelan Parameter Terbaik

Setelah skenario dibentuk, setiap skenario diekstraksi menjadi vektor fitur yang mewakili karakteristik *workload*. Fitur yang digunakan meliputi jumlah *job*, total ukuran *workload*, rata-rata ukuran

job, simpangan baku ukuran *job*, koefisien variasi, kuantil ukuran *job*, proporsi *job* kecil dan *job* besar, beban kerja per VM, serta ukuran-ukuran yang berkaitan dengan tightness deadline. Selain itu, ditambahkan pula fitur regime dalam bentuk numerik dan beberapa fitur interaksi untuk membantu model membedakan skenario dengan distribusi beban kerja yang mirip tetapi tingkat keketatannya berbeda.

Tahap berikutnya adalah membentuk label parameter terbaik untuk *genetic algorithm*. Pada setiap skenario, terlebih dahulu dihitung performa *baseline* FCFS dan EDF sebagai dasar normalisasi. Selanjutnya, sejumlah kombinasi parameter *genetic algorithm* dievaluasi untuk memperoleh konfigurasi dengan nilai objektif terbaik. Parameter yang dituning meliputi *crossover rate*, *mutation rate*, *elitism fraction*, dan *tournament size*. Nilai objektif gabungan dirumuskan secara eksplisit sebagai berikut:

$$F = \alpha \cdot M_{norm} + \beta \cdot P_{norm}$$

dengan:

(1)

$$M_{norm} = \frac{M}{M_{baseline}}$$
$$P_{norm} = \frac{P}{P_{baseline}}$$

Keterangan:

F adalah nilai objektif total, M_{norm} adalah *makespan* yang telah dinormalisasi, P_{norm} adalah *total penalty* yang telah dinormalisasi, α adalah bobot *makespan*, dan β adalah bobot *penalty*. Pada penelitian ini digunakan $\alpha = 0,4$ dan $\beta = 0,6$, sehingga fungsi objektif dibuat lebih sensitif terhadap *penalty*. Kombinasi parameter yang menghasilkan nilai F terkecil pada suatu skenario ditetapkan sebagai label parameter terbaik.

Pemilihan formulasi ini didasarkan pada fakta bahwa penjadwalan cloud modern tidak hanya berupaya mempersingkat waktu penyelesaian, tetapi juga mengendalikan biaya keterlambatan dan konsekuensi kualitas layanan [3], [26], [32], [33]. Kombinasi parameter yang menghasilkan nilai J terkecil pada suatu skenario ditetapkan sebagai label parameter terbaik. Dengan demikian, proses tuning parameter diformulasikan sebagai masalah supervised learning, yaitu mempelajari pemetaan dari fitur *workload* menuju konfigurasi parameter GA yang paling efektif untuk skenario tersebut.

2.4 Model Prediksi Parameter dan Metode Pembandingan

Metode usulan pada penelitian ini adalah GA-ML, yaitu pendekatan tuning parameter *genetic algorithm* berbantuan *machine learning*. Tiga parameter kontinu, yaitu *crossover rate*, *mutation rate*, dan *elitism fraction*, diprediksi menggunakan model regresi. Sementara itu, parameter diskrit *tournament size* diprediksi menggunakan model klasifikasi. Pada penelitian ini, model *machine learning* yang digunakan adalah RandomForestRegressor untuk memprediksi tiga parameter kontinu, yaitu *crossover rate*, *mutation rate*, dan *elitism fraction*. Sementara itu, RandomForestClassifier digunakan untuk memprediksi parameter diskrit, yaitu *tournament size*. Kedua model diimplementasikan menggunakan pustaka Scikit-learn pada bahasa pemrograman Python. Pemilihan Random Forest didasarkan pada kemampuannya menangani hubungan nonlinier antara fitur *workload* dan parameter GA, ketahanannya terhadap variasi data, serta kesesuaiannya untuk dataset berukuran kecil hingga menengah tanpa memerlukan asumsi distribusi tertentu.

Konfigurasi model yang digunakan dibuat eksplisit untuk mendukung reproduktibilitas. Pada RandomForestRegressor, parameter yang digunakan meliputi `n_estimators = 300`, `max_depth = 8`, `min_samples_leaf = 1`, `min_samples_split = 2`, `max_features = sqrt`, `bootstrap = True`, `random_state =`

42, dan $n_jobs = -1$. Pada RandomForestClassifier, parameter yang digunakan sama, dengan tambahan `class_weight = balanced` untuk mengurangi bias pada kelas parameter diskrit. Model dilatih menggunakan data latih yang berisi fitur karakteristik *workload* dan label parameter terbaik hasil pencarian GA. Setelah prediksi dilakukan, nilai parameter kontinu disesuaikan ke kandidat parameter terdekat yang valid, sedangkan parameter *tournament size* disesuaikan ke nilai kandidat diskrit yang tersedia.

Setelah parameter diprediksi, penelitian ini menambahkan local refinement ringan di sekitar nilai prediksi untuk mencari konfigurasi yang paling menjanjikan pada lingkungan terdekat. Dengan demikian, *machine learning* digunakan sebagai pengarah awal pencarian, sedangkan pencarian lokal mempertajam parameter akhir sebelum *scheduler* dijalankan. Pendekatan ini sejalan dengan gagasan bahwa *machine learning* dapat memperkuat proses pencarian metaheuristik tanpa harus mengganti mekanisme optimasi utamanya [30], [31]. Untuk menilai efektivitas metode usulan, digunakan empat metode pembandingan:

1. FCFS, sebagai *baseline* penjadwalan sederhana berbasis urutan kedatangan *job*.
2. EDF, sebagai *baseline deadline-aware* yang memprioritaskan *job* dengan deadline terdekat.
3. GA-Default, yaitu *genetic algorithm* dengan parameter tetap yang sama untuk seluruh skenario.
4. GA-Grid, yaitu *genetic algorithm* dengan parameter global terbaik hasil pencarian pada data latih.

Pemilihan pembandingan ini dimaksudkan agar evaluasi dapat menjawab dua pertanyaan sekaligus, yaitu apakah pendekatan metaheuristik lebih baik daripada *baseline* klasik, dan apakah tuning parameter berbantuan *machine learning* lebih baik daripada konfigurasi tetap maupun tuning grid konvensional [13], [15], [23], [30], [31].

2.5 Prosedur Eksperimen

Data skenario dibagi menjadi data latih dan data uji dengan proporsi 70:30. Data latih digunakan untuk membangun model prediksi parameter dan untuk menentukan konfigurasi global terbaik pada metode GA-Grid. Selanjutnya, data uji digunakan untuk mengevaluasi seluruh metode pembandingan pada skenario yang tidak dilibatkan selama proses pelatihan. Agar hasil evaluasi lebih stabil, eksperimen akhir dijalankan dengan skema *multi-seed*, yaitu tiga seed independen. Pada setiap seed, metode GA-Default, GA-Grid, dan GA-ML dijalankan kembali, kemudian hasilnya dirata-ratakan. Strategi ini digunakan karena genetic algorithm bersifat stokastik, sehingga satu kali eksekusi dapat menghasilkan variasi performa yang cukup besar. Dengan demikian, evaluasi multi-seed memberikan gambaran yang lebih adil terhadap konsistensi performa masing-masing metode.

Untuk mengurangi risiko *overfitting* akibat jumlah skenario yang relatif terbatas, evaluasi model prediksi parameter juga diperkuat menggunakan Repeated K-Fold Cross-Validation. Berdasarkan program eksperimen, digunakan 5-fold cross-validation dengan 5 pengulangan, sehingga diperoleh 25 evaluasi lipatan untuk setiap parameter. Validasi silang ini digunakan untuk memastikan bahwa performa prediksi parameter tidak hanya bergantung pada satu kali pembagian data latih dan data uji. Selain itu, pembagian data utama tetap dilakukan dengan proporsi 70:30, menghasilkan 31 skenario sebagai data latih dan 14 skenario sebagai data uji.

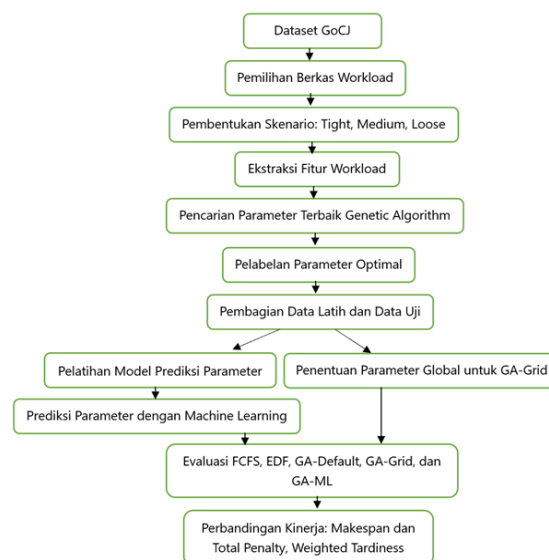
Meskipun jumlah skenario adalah 45, skenario tersebut tidak berasal dari satu kondisi yang homogen, melainkan dibentuk dari 15 berkas *workload* mentah GoCJ dan tiga *regime* berbeda, yaitu *tight*, *medium*, dan *loose*. Dengan demikian, data tetap merepresentasikan variasi jumlah *job*, ukuran *workload*, distribusi ukuran *job*, tingkat keketatan *deadline*, serta beban kerja per VM. Oleh karena itu, dataset ini dinilai memadai untuk studi awal mengenai *machine learning-assisted parameter tuning*, meskipun pengujian pada dataset dan jumlah skenario yang lebih besar tetap diperlukan untuk memperkuat generalisasi hasil.

Secara umum, prosedur eksperimen pada penelitian ini terdiri atas beberapa tahap. Pertama, berkas workload mentah GoCJ dibaca dan dipilih untuk membentuk himpunan data eksperimen. Kedua, setiap workload dibentuk menjadi tiga skenario, yaitu *tight*, *medium*, dan *loose*. Ketiga, dari setiap skenario diekstraksi fitur-fitur workload yang digunakan sebagai masukan model machine learning. Keempat, dilakukan pencarian parameter genetic algorithm untuk memperoleh label parameter terbaik berdasarkan fungsi objektif yang digunakan. Kelima, data dibagi menjadi data latih dan data uji. Keenam, model prediksi parameter dilatih menggunakan data latih. Ketujuh, parameter global terbaik untuk metode GA-Grid ditentukan pada data latih. Kedelapan, metode FCFS, EDF, GA-Default, GA-Grid, dan GA-ML dijalankan pada data uji. Kesembilan, hasil seluruh metode dievaluasi dan dibandingkan berdasarkan metrik performa yang telah ditentukan. Alur umum penelitian ini ditunjukkan pada Gambar 1.

2.6 Analisis overhead komputasi.

Dari sisi overhead komputasi, pendekatan GA-ML memiliki tambahan proses berupa ekstraksi fitur *workload*, prediksi parameter menggunakan model *machine learning*, dan *local refinement* ringan di sekitar parameter hasil prediksi. Namun, proses pelatihan model dilakukan secara *offline* menggunakan data latih, sehingga tidak menambah beban komputasi pada tahap penjadwalan utama. Pada tahap pengujian, overhead GA-ML terutama berasal dari proses prediksi parameter dan pencarian lokal terbatas, bukan dari pencarian kombinasi parameter secara menyeluruh.

Dibandingkan GA-Grid, GA-ML lebih praktis karena GA-Grid membutuhkan pencarian konfigurasi parameter berbasis kombinasi kandidat pada data latih, sedangkan GA-ML menggunakan model prediktif untuk mengarahkan pemilihan parameter berdasarkan karakteristik *workload*. Dengan demikian, meskipun GA-ML menambahkan komponen prediktif, overhead tersebut masih dapat diterima karena proses tuning menjadi lebih terarah dan tidak perlu melakukan pencarian grid penuh pada setiap skenario uji.



Gambar 1. Alur umum penelitian machine learning-assisted parameter tuning untuk metaheuristic cloud scheduling

2.7 Metrik Evaluasi

Evaluasi dilakukan pada dua level, yaitu akurasi prediksi parameter dan dampak terhadap performa penjadwalan. Untuk akurasi prediksi parameter kontinu digunakan Mean Absolute Error (MAE) dan Root Mean Squared Error (RMSE), yang dirumuskan sebagai:

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (2)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} \quad (3)$$

dengan:

- y_i = nilai parameter aktual,
- $\hat{y}_i$ = nilai parameter hasil prediksi,
- n = jumlah sampel.

Untuk mengevaluasi dampak terhadap penjadwalan, digunakan empat metrik utama, yaitu makespan, total penalty, weighted tardiness, dan violation rate. Makespan menunjukkan total waktu penyelesaian seluruh *job* pada suatu skenario. Total penalty menyatakan akumulasi penalti keterlambatan. Weighted tardiness digunakan untuk mengukur keterlambatan dengan mempertimbangkan bobot penalti masing-masing *job*. Rumus *weighted tardiness* dinyatakan sebagai berikut.:

$$WT = \sum_{i=1}^n w_i \cdot \max(0, C_i - d_i) \quad (4)$$

dengan:

- WT = *weighted tardiness*,
- w_i = bobot penalti *job* ke- i ,
- C_i = waktu selesai *job* ke- i ,
- d_i = *deadline* *job* ke- i .

Adapun violation rate dihitung sebagai proporsi *job* yang selesai melewati *deadline*. Rumusnya dinyatakan sebagai berikut.

$$VR = \frac{1}{n} \sum_{i=1}^n I(C_i > d_i) \quad (5)$$

dengan:

- VR = violation rate,
- $I(C_i > d_i)$ = fungsi indikator yang bernilai 1 jika *job* selesai melewati *deadline* dan 0 jika tidak,
- n = jumlah *job* dalam skenario.

Kombinasi metrik tersebut dipilih karena mampu menunjukkan dua sisi sekaligus, yaitu kualitas prediksi parameter dan konsekuensi operasionalnya pada proses penjadwalan. Dengan demikian, metode usulan tidak hanya dinilai dari ketepatan memprediksi parameter GA, tetapi juga dari sejauh mana parameter tersebut benar-benar meningkatkan kinerja scheduling dibandingkan metode pembandingan [7], [13], [25], [26].

2.8 Ruang Lingkup dan Keterbatasan Metode

Penelitian ini memiliki beberapa batasan. Pertama, dataset dasar yang digunakan adalah dataset mentah GoCJ, sehingga atribut *deadline* dan *penalty* dibentuk secara eksperimental dan bukan diperoleh langsung dari dataset asli. Kedua, metaheuristik yang digunakan dalam penelitian ini difokuskan pada *genetic algorithm*, sehingga generalisasi hasil ke metaheuristik lain masih memerlukan pengujian lanjutan. Ketiga, *machine learning* pada penelitian ini diposisikan sebagai mekanisme assisted tuning, bukan sebagai *scheduler* otonom penuh. Oleh karena itu, kontribusi utama penelitian terletak pada otomatisasi pemilihan parameter, bukan pada penggantian total pipeline penjadwalan.

3. Hasil dan Pembahasan

Bagian ini menyajikan hasil evaluasi metode usulan GA-ML terhadap beberapa metode pembandingan, yaitu FCFS, EDF, GA-Default, dan GA-Grid. Evaluasi dilakukan pada data uji yang dibentuk

dari skenario *workload* berbasis GoCJ. Pembahasan difokuskan pada dua aspek utama, yaitu ketepatan model *machine learning* dalam memprediksi parameter *genetic algorithm* dan dampak parameter tersebut terhadap performa penjadwalan cloud.

3.1. Hasil Prediksi Parameter Genetic Algorithm

Tahap pertama evaluasi dilakukan untuk menilai kemampuan model *machine learning* dalam memprediksi parameter *genetic algorithm* terbaik berdasarkan karakteristik *workload*. Parameter yang diprediksi meliputi *crossover rate*, *mutation rate*, *elitism fraction*, dan *tournament size*. Hasil evaluasi akurasi prediksi tersebut disajikan pada Tabel 1.

Tabel 1. Akurasi Prediksi Parameter Genetic Algorithm oleh Model Machine Learning

Parameter	RMSE	MAE
best_crossover_rate	0,122	0,093
best_mutation_rate	0,032	0,026
best_elitism_frac	0,030	0,018
best_tournament_size	0,964	0,786

Berdasarkan Tabel 1, model *machine learning* menunjukkan kemampuan prediksi yang cukup baik untuk parameter kontinu. Nilai *error* terendah diperoleh pada *elitism fraction* dengan RMSE 0,030 dan MAE 0,018, diikuti oleh *mutation rate* dengan RMSE 0,032 dan MAE 0,026. Hasil ini menunjukkan bahwa pola hubungan antara karakteristik *workload* dan kedua parameter tersebut dapat dipelajari dengan cukup stabil oleh model. Untuk *crossover rate*, tingkat kesalahan prediksi masih tergolong wajar dengan RMSE 0,122 dan MAE 0,093. Meskipun lebih tinggi dibandingkan dua parameter kontinu lainnya, nilai ini tetap menunjukkan bahwa model mampu mendekati konfigurasi *crossover* yang relevan untuk berbagai skenario *workload*. Sementara itu, *tournament size* menjadi parameter yang paling sulit diprediksi, dengan RMSE 0,964 dan MAE 0,786. Kondisi ini dapat dipahami karena *tournament size* merupakan parameter diskrit dengan rentang nilai yang sempit, sehingga perubahan satu tingkat nilai saja dapat langsung meningkatkan *error* numeriknya. Secara umum, hasil pada Tabel 1 menunjukkan bahwa pendekatan *machine learning* yang digunakan sudah memadai untuk memprediksi konfigurasi parameter *genetic algorithm*, terutama untuk parameter kontinu. Dengan demikian, model prediktif yang dibangun dapat digunakan sebagai dasar untuk mendukung proses *tuning* parameter pada tahap penjadwalan.

Tabel 1a. Hasil Repeated K-Fold Cross-Validation Prediksi Parameter GA

Parameter	Model	RMSE	RMSE	MAE	MAE	Jumlah Fold
		Mean	Std	Mean	Std	
best_crossover_rate	RandomForestRegressor	0,105	0,028	0,078	0,027	25
best_elitism_frac	RandomForestRegressor	0,046	0,009	0,036	0,010	25
best_mutation_rate	RandomForestRegressor	0,027	0,005	0,021	0,005	25
best_tournament_size	RandomForestClassifier	1,333	0,246	1,036	0,274	25

Berdasarkan Tabel 1a, hasil *Repeated K-Fold Cross-Validation* menunjukkan bahwa model mampu mempertahankan *error* prediksi yang relatif stabil pada sebagian besar parameter kontinu. Nilai *error* terendah diperoleh pada *mutation rate* dan *elitism fraction*, sedangkan *tournament size* memiliki *error* lebih tinggi karena parameter tersebut bersifat diskrit dan memiliki rentang kandidat terbatas. Hasil ini menunjukkan bahwa model prediksi parameter tidak hanya bekerja pada satu pembagian data, tetapi juga relatif stabil pada validasi silang berulang.

3.2. Perbandingan Kinerja Metode Penjadwalan

Setelah parameter diprediksi, tahap berikutnya adalah mengevaluasi dampaknya terhadap performa penjadwalan. Ringkasan hasil rata-rata kinerja seluruh metode pada data uji disajikan pada

Tabel 2. Seluruh metrik pada tabel tersebut bersifat *lower is better*, sehingga nilai yang lebih kecil menunjukkan performa yang lebih baik.

Tabel 2. Perbandingan Kinerja Metode Penjadwalan pada Data Uji

Metode	Makespan	Total Penalty	Weighted Tardiness	Violation Rate
FCFS	1750,214	936577,736	1643,539	0,901
EDF	1816,418	690286,925	1192,039	0,902
GA-Default	1627,286	814784,916	1410,799	0,903
GA-Grid	1622,620	809872,019	1403,499	0,905
GA-ML	1632,067	742901,557	1279,938	0,892

Berdasarkan Tabel 2, pada metrik makespan, metode GA-Grid menghasilkan nilai terendah, yaitu 1622,620, diikuti oleh GA-Default sebesar 1627,286 dan GA-ML sebesar 1632,067. Dengan demikian, GA-ML belum menjadi metode terbaik untuk metrik makespan karena nilainya masih sedikit lebih tinggi dibandingkan GA-Grid dan GA-Default. Namun, GA-ML tetap menghasilkan makespan yang lebih rendah dibandingkan FCFS sebesar 1750,214 dan EDF sebesar 1816,418. Hasil ini menunjukkan bahwa metode berbasis genetic algorithm secara umum lebih efektif dalam menekan waktu penyelesaian total dibandingkan baseline klasik, meskipun pada aspek makespan GA-Grid masih menjadi metode terbaik.

Pada metrik total penalty, EDF menghasilkan nilai terendah secara keseluruhan, yaitu 690286,925. Hal ini menunjukkan bahwa EDF masih sangat kuat untuk skenario yang sensitif terhadap deadline. Namun, di antara metode berbasis genetic algorithm, GA-ML menunjukkan kinerja terbaik dengan total penalty sebesar 742901,557. Nilai ini lebih rendah dibandingkan GA-Grid sebesar 809872,019 dan GA-Default sebesar 814784,916. Dibandingkan GA-Grid, GA-ML mampu menurunkan total penalty sebesar 8,269%. Temuan ini menunjukkan bahwa penyetelan parameter berbantuan machine learning mampu meningkatkan kualitas genetic algorithm dalam menekan akumulasi penalti akibat keterlambatan.

Pola serupa juga terlihat pada metrik weighted tardiness. EDF masih menghasilkan nilai terendah secara keseluruhan, yaitu 1192,039. Akan tetapi, dalam kelompok metode berbasis genetic algorithm, GA-ML memperoleh hasil terbaik dengan nilai weighted tardiness sebesar 1279,938. Nilai ini lebih rendah dibandingkan GA-Grid sebesar 1403,499 dan GA-Default sebesar 1410,799. Berdasarkan Tabel 3, GA-ML memberikan peningkatan sebesar 8,804% dibandingkan GA-Grid. Hasil ini menunjukkan bahwa parameter yang diprediksi oleh machine learning mampu menghasilkan konfigurasi GA yang lebih efektif dalam menekan keterlambatan berbobot.

Tabel 3. Ringkasan Uji Statistik GA-ML terhadap GA-Grid

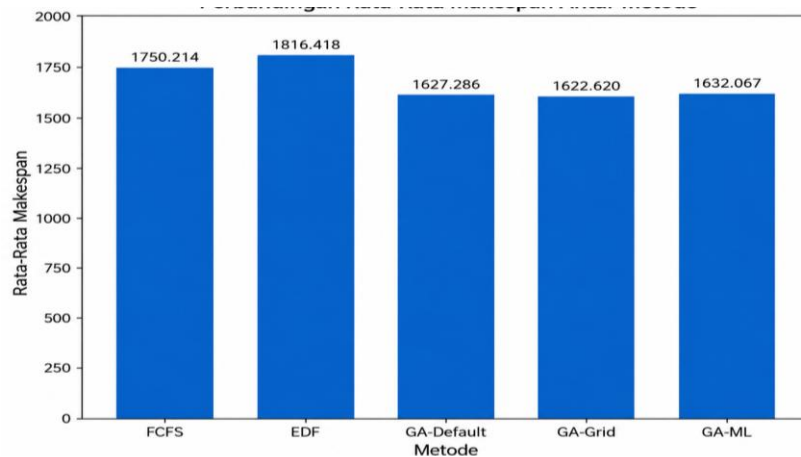
Metrik	Improvement GA-ML	p-value	Metode Lebih Baik
Makespan	-0,582%	0,035278	GA-Grid
Total Penalty	8,269%	0,000122	GA-ML
Weighted Tardiness	8,804%	0,000122	GA-ML
Violation Rate	1,366%	0,000610	GA-ML

Dengan demikian, meskipun selisih numerik antara GA-ML dan GA-Grid relatif kecil pada beberapa metrik, hasil tersebut tetap memiliki signifikansi statistik dan signifikansi praktis, terutama karena perbaikan terjadi pada metrik yang berkaitan langsung dengan kualitas layanan, yaitu *total penalty*, *weighted tardiness*, dan *violation rate*.

Pada metrik violation rate (Tabel 2), GA-ML menghasilkan nilai terbaik dibandingkan seluruh metode, yaitu 0,892. Nilai ini lebih rendah dibandingkan FCFS sebesar 0,901, EDF sebesar 0,902, GA-Default sebesar 0,903, dan GA-Grid sebesar 0,905. Dibandingkan GA-Grid, GA-ML mampu menurunkan violation rate sebesar 1,366%. Hasil ini penting karena menunjukkan bahwa GA-ML tidak hanya mampu menekan penalty dan weighted tardiness, tetapi juga mampu mengurangi proporsi job yang melewati deadline.

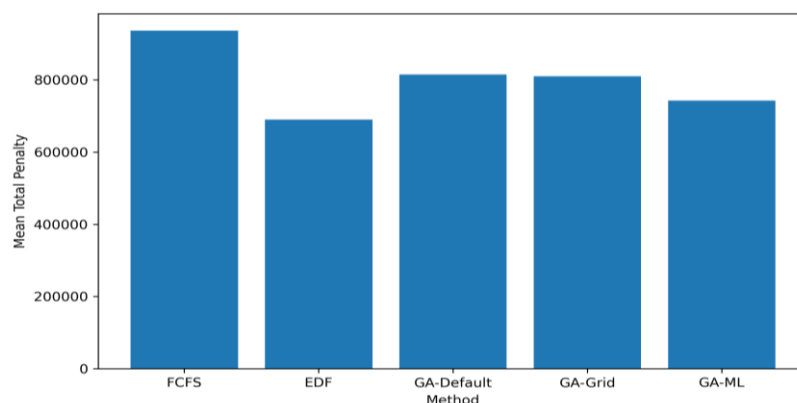
Tabel 3 menunjukkan hasil perbandingan langsung antara GA-ML dan GA-Grid menggunakan uji Wilcoxon signed-rank. Seluruh nilai p-value berada di bawah tingkat signifikansi 0,05, sehingga perbedaan antara GA-ML dan GA-Grid signifikan pada seluruh metrik evaluasi. GA-ML secara signifikan lebih baik daripada GA-Grid pada tiga metrik, yaitu total penalty, weighted tardiness, dan violation rate. Namun, pada metrik makespan, GA-ML mengalami penurunan performa sebesar 0,582%, sehingga GA-Grid masih lebih unggul untuk minimisasi waktu penyelesaian total. Dengan demikian, hasil ini menunjukkan adanya trade-off kecil pada makespan, tetapi GA-ML memberikan keuntungan yang lebih kuat pada metrik berbasis SLA.

Untuk memperjelas perbedaan antar-metode pada metrik utama, Gambar 2 menampilkan rata-rata makespan setiap metode pada data uji.



Gambar 2. Rata-rata Makespan Setiap Metode pada Data Uji

Berdasarkan Gambar 2, terlihat bahwa EDF menghasilkan makespan tertinggi, sedangkan GA-Grid menghasilkan makespan terendah. GA-ML memiliki makespan sedikit lebih tinggi dibandingkan GA-Grid dan GA-Default, tetapi masih lebih rendah dibandingkan FCFS dan EDF. Temuan ini menunjukkan bahwa metode berbasis genetic algorithm lebih efektif dibandingkan baseline klasik dalam menekan makespan. Namun, untuk metrik makespan secara khusus, GA-Grid masih menjadi metode terbaik, sedangkan GA-ML menunjukkan trade-off kecil karena lebih diarahkan pada perbaikan metrik berbasis SLA. Perbandingan kinerja metode dalam menekan total penalty ditunjukkan pada Gambar 3.

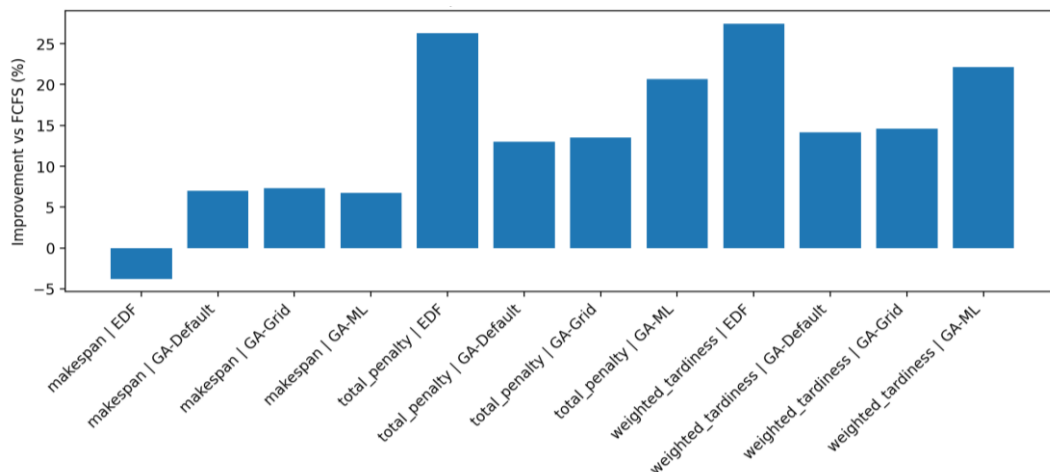


Gambar 3. Rata-rata Total Penalty Setiap Metode pada Data Uji

Pada Gambar 3, EDF menghasilkan total penalty paling rendah di antara seluruh metode. Hal ini menunjukkan bahwa EDF tetap kuat untuk skenario penjadwalan yang sangat sensitif terhadap deadline. Meskipun demikian, di antara metode berbasis genetic algorithm, GA-ML menghasilkan total

penalty paling rendah. Dengan demikian, machine learning-assisted tuning berhasil meningkatkan performa genetic algorithm dibandingkan GA-Default dan GA-Grid dalam menekan penalti keterlambatan. Untuk melihat keuntungan relatif terhadap baseline FCFS, persentase peningkatan kinerja setiap metode disajikan pada Gambar 4

Berdasarkan Gambar 4, GA-ML memberikan peningkatan sebesar 6,750% pada makespan, 20,679% pada total penalty, dan 22,123% pada weighted tardiness dibandingkan FCFS. Selain itu, GA-ML juga mampu menurunkan violation rate sebesar 0,957% terhadap FCFS. Meskipun peningkatan makespan GA-ML tidak sebesar GA-Grid, GA-ML memberikan keuntungan yang lebih kuat pada total penalty, weighted tardiness, dan violation rate. Dengan demikian, pendekatan prediktif yang diusulkan lebih sesuai untuk skenario penjadwalan cloud yang menekankan kualitas layanan, pengurangan penalti, keterlambatan berbobot, dan pelanggaran deadline.



Gambar 4. Persentase Peningkatan Kinerja terhadap FCFS pada Data Uji

3.3. Pembahasan

Secara keseluruhan, hasil eksperimen menunjukkan bahwa pendekatan penyetelan parameter berbantuan *machine learning* untuk penjadwalan *cloud* berbasis metaheuristik mampu menghasilkan performa yang kompetitif pada skenario *cloud scheduling* berbasis *workload* GoCJ. Temuan utama penelitian ini adalah bahwa *machine learning* dapat dimanfaatkan untuk mengotomatisasi pemilihan parameter *genetic algorithm* berdasarkan karakteristik *workload*, sehingga konfigurasi parameter tidak hanya bergantung pada satu nilai tetap untuk seluruh skenario. Hal ini penting karena lingkungan *cloud* bersifat dinamis, sehingga parameter yang efektif pada satu kondisi *workload* belum tentu tetap optimal pada kondisi lainnya.

Berdasarkan hasil pada Tabel 2, metode berbasis *genetic algorithm* menunjukkan performa yang lebih baik dibandingkan baseline klasik pada metrik *makespan*. GA-Grid memperoleh nilai *makespan* terendah, yaitu 1622,620, diikuti oleh GA-Default sebesar 1627,286 dan GA-ML sebesar 1632,067. Dengan demikian, GA-ML belum menjadi metode terbaik untuk metrik *makespan*. Namun, GA-ML tetap menghasilkan *makespan* yang lebih rendah dibandingkan FCFS dan EDF. Temuan ini menunjukkan bahwa pendekatan berbasis GA efektif untuk menekan waktu penyelesaian total, meskipun tuning berbantuan *machine learning* belum memberikan hasil terbaik pada metrik tersebut.

Pada metrik *total penalty*, EDF menghasilkan nilai terbaik secara keseluruhan, yaitu 690286,925. Hal ini menunjukkan bahwa EDF masih sangat kuat pada skenario penjadwalan yang sensitif terhadap *deadline*. Namun, di antara metode berbasis *genetic algorithm*, GA-ML menghasilkan performa terbaik dengan nilai *total penalty* sebesar 742901,557. Nilai ini lebih rendah dibandingkan GA-Grid sebesar

809872,019 dan GA-Default sebesar 814784,916. Berdasarkan Tabel 3, GA-ML mampu menurunkan *total penalty* sebesar 8,269% dibandingkan GA-Grid, dan perbedaan tersebut signifikan berdasarkan uji Wilcoxon *signed-rank*. Hal ini menunjukkan bahwa parameter yang diprediksi oleh *machine learning* mampu memperbaiki kualitas GA dalam menekan akumulasi penalti akibat keterlambatan.

Pola yang sama terlihat pada metrik *weighted tardiness*. EDF tetap menjadi metode terbaik secara keseluruhan dengan nilai 1192,039. Akan tetapi, dalam keluarga metode berbasis *genetic algorithm*, GA-ML memperoleh nilai terbaik sebesar 1279,938, lebih rendah dibandingkan GA-Grid sebesar 1403,499 dan GA-Default sebesar 1410,799. Dibandingkan GA-Grid, GA-ML memberikan peningkatan sebesar 8,804%, dan hasil tersebut signifikan secara statistik. Temuan ini mengindikasikan bahwa penyetelan parameter berbantuan *machine learning* mampu menghasilkan konfigurasi GA yang lebih efektif dalam menekan keterlambatan berbobot.

Hasil yang lebih menarik terlihat pada metrik *violation rate*. Berbeda dari hasil sebelumnya, pada hasil terbaru GA-ML menunjukkan keunggulan paling baik dibandingkan seluruh metode, dengan nilai *violation rate* sebesar 0,892. Nilai ini lebih rendah dibandingkan FCFS sebesar 0,901, EDF sebesar 0,902, GA-Default sebesar 0,903, dan GA-Grid sebesar 0,905. Dibandingkan GA-Grid, GA-ML mampu menurunkan *violation rate* sebesar 1,366%, dan perbedaan tersebut signifikan secara statistik. Dengan demikian, metode usulan tidak hanya efektif dalam menekan penalti dan keterlambatan berbobot, tetapi juga mampu mengurangi proporsi job yang melewati *deadline*.

Hasil uji statistik pada Tabel 3 memperkuat interpretasi tersebut. Seluruh nilai *p-value* berada di bawah tingkat signifikansi 0,05, sehingga perbedaan antara GA-ML dan GA-Grid signifikan pada seluruh metrik evaluasi. GA-ML lebih unggul secara signifikan pada tiga dari empat metrik, yaitu *total penalty*, *weighted tardiness*, dan *violation rate*. Namun, pada metrik *makespan*, GA-ML mengalami penurunan performa sebesar 0,582% dibandingkan GA-Grid. Hal ini menunjukkan adanya *trade-off* kecil antara minimisasi waktu penyelesaian total dan peningkatan performa pada metrik berbasis SLA.

Dari sudut pandang metodologis, hasil ini menunjukkan bahwa *machine learning-assisted tuning* dapat menjadi alternatif yang lebih adaptif dibandingkan penggunaan parameter tetap maupun tuning grid konvensional. GA-ML tidak selalu menghasilkan nilai terbaik pada seluruh metrik, tetapi mampu meningkatkan kualitas keputusan GA pada metrik yang lebih relevan dengan SLA, yaitu penalti, keterlambatan berbobot, dan pelanggaran *deadline*. Dengan kata lain, keunggulan utama GA-ML bukan terletak pada dominasi absolut terhadap seluruh metode pembandingan, melainkan pada kemampuannya mengarahkan konfigurasi parameter GA agar lebih sesuai dengan karakteristik *workload* dan tujuan penjadwalan berbasis kualitas layanan.

Meskipun demikian, hasil penelitian juga menunjukkan bahwa EDF masih menjadi pembandingan yang kuat pada metrik yang sangat sensitif terhadap *deadline*, khususnya *total penalty* dan *weighted tardiness*. Sementara itu, GA-Grid tetap lebih unggul pada metrik *makespan*. Oleh karena itu, interpretasi hasil harus dilakukan secara proporsional. GA-ML lebih tepat diposisikan sebagai metode yang efektif untuk meningkatkan performa GA pada metrik berbasis SLA, bukan sebagai metode yang selalu terbaik pada semua metrik.

Dengan demikian, kontribusi penelitian ini terletak pada formulasi tuning parameter metaheuristik sebagai masalah prediksi berbasis *machine learning* serta pembuktian empiris bahwa pendekatan tersebut mampu meningkatkan performa GA dibandingkan tuning grid konvensional pada tiga metrik penting. Hasil ini membuka peluang pengembangan lebih lanjut, misalnya dengan memperkaya fitur *workload*, memperluas pendekatan ke metaheuristik lain, serta merancang fungsi objektif yang lebih seimbang antara minimisasi *makespan*, pengurangan penalti, keterlambatan berbobot, dan *deadline violation*.

4. Kesimpulan

Penelitian ini mengusulkan pendekatan penyetelan parameter berbantuan *machine learning* untuk penjadwalan *cloud* berbasis *genetic algorithm*. Pendekatan ini memanfaatkan karakteristik *workload* untuk memprediksi konfigurasi parameter GA, sehingga proses tuning tidak lagi bergantung pada parameter tetap maupun pencarian grid konvensional.

Hasil eksperimen menunjukkan bahwa GA-ML mampu memberikan kinerja kompetitif dibandingkan metode pembandingan. Dibandingkan FCFS, GA-ML menurunkan *makespan* sebesar 6,750%, *total penalty* sebesar 20,679%, *weighted tardiness* sebesar 22,123%, dan *violation rate* sebesar 0,957%. Dibandingkan GA-Grid, GA-ML unggul pada tiga dari empat metrik, yaitu menurunkan *total penalty* sebesar 8,269%, *weighted tardiness* sebesar 8,804%, dan *violation rate* sebesar 1,366%. Seluruh perbedaan tersebut signifikan berdasarkan uji Wilcoxon *signed-rank* pada tingkat signifikansi 0,05.

Meskipun demikian, GA-ML belum unggul pada metrik *makespan*, karena GA-Grid menghasilkan nilai *makespan* yang sedikit lebih rendah. Dengan demikian, GA-ML lebih tepat diposisikan sebagai metode yang kuat untuk penjadwalan berbasis SLA, terutama dalam menekan penalti, keterlambatan berbobot, dan pelanggaran *deadline*, meskipun terdapat *trade-off* kecil pada waktu penyelesaian total. Untuk penelitian selanjutnya, pendekatan ini dapat dikembangkan dengan memperkaya fitur *workload*, menguji metaheuristik lain, serta merancang fungsi objektif yang lebih seimbang antara *makespan* dan metrik berbasis SLA.

Pustaka

- [1] W. K. Awad, Khairul Akram Zainol Ariffin, M. Z. A. Nazri, and E. T. Yassen, "Resource allocation strategies and task scheduling algorithms in cloud computing: a systematic literature review," *J. Intell. Syst.*, vol. 34, no. 1, 2025, doi: 10.1515/jisys-2024-0441.
- [2] N. Devi *et al.*, "A systematic literature review for load balancing and task scheduling techniques in cloud computing," *Artif. Intell. Rev.*, vol. 57, no. 10, p. 276, 2024, doi: 10.1007/s10462-024-10925-w.
- [3] S. H. H. Madni, M. Faheem, M. Younas, M. H. Masum, and S. Shah, "Critical review on resource scheduling in IaaS clouds: Taxonomy, issues, challenges, and future directions," *J. Eng.*, no. 8, p. e12420, 2024, doi: 10.1049/tje2.12420.
- [4] W. Khallouli and J. Huang, "Cluster resource scheduling in cloud computing: literature review and research challenges," *J. Supercomput.*, vol. 78, no. 5, pp. 6898–6943, 2022, doi: 10.1007/s11227-021-04138-z.
- [5] R. Aron and A. Abraham, "Resource scheduling methods for cloud computing environment: The role of meta-heuristics and artificial intelligence," *Eng. Appl. Artif. Intell.*, vol. 116, no. September, p. 105345, 2022, doi: 10.1016/j.engappai.2022.105345.
- [6] D. Zhang, Xiaolan Xie, and Y. Song, "A Multi-Objective Optimization-Based Container Cloud Scheduling Method for Multi-Scenario Resource Management," *Futur. Internet*, vol. 18, no. 1, p. 58, 2026, doi: 10.3390/fi18010058.
- [7] J. Pan, Yi Wei, L. Meng, and X. Meng, "A dual scheduling framework for task and resource allocation in cloud computing," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 37, no. 5, p. 81, 2025, doi: 10.1007/s44443-025-00092-5.
- [8] L. Mao, R. Chen, H. Cheng, W. Lin, B. Liu, and J. Z. Wang, "A resource scheduling method for cloud data centers based on thermal management," *J. Cloud Comput.*, vol. 12, no. 84, 2023, doi:

10.1186/s13677-023-00462-2.

- [9] S. S. Sefati *et al.*, "Adaptive Resource Scheduling in Multi-Cloud Computing Using Recurrent Neural Forecasting and Memory-Based Metaheuristic Optimization," *J. Grid Comput.*, vol. 23, no. 4, p. 26, 2025, doi: 10.1007/s10723-025-09812-7.
- [10] H.-M. Song *et al.*, "Task scheduling of cloud computing system by frilled lizard optimization with time varying expansion mixed function oscillation and horned lizard camouflage strategy," *J. Netw. Comput. Appl.*, vol. 245, p. 104386, 2025, doi: 10.1016/j.jnca.2025.104386.
- [11] Y.-F. Sun, Si-Wen Zhang, J.-S. Wang, S.-H. Zhang, Y.-C. Wang, and X.-F. Sui, "Task scheduling in cloud computing system by improved honey badger optimization algorithm with two dimensional and three dimensional fractals," *Sustain. Comput. Informatics Syst.*, vol. 48, p. 101201, 2025, doi: 10.1016/j.suscom.2025.101201.
- [12] Y. Han, "Cloud computing task scheduling based on particle swarm optimization algorithm," *Procedia Comput. Sci.*, vol. 261, pp. 1349–1355, 2025, doi: 10.1016/j.procs.2025.05.012.
- [13] K. J. S and G. M., "A Probabilistic Genetic Algorithm Approach to Efficient Task Scheduling in Cloud Environments," *CLEI Electron. J.*, vol. 28, no. 5, 2025, doi: 10.19153/cleiej.28.5.9.
- [14] X. Zhang, "Optimizing scientific workflow scheduling in cloud computing: a multi-level approach using whale optimization algorithm," *J. Eng. Appl. Sci.*, vol. 71, no. 1, p. 175, 2024, doi: 10.1186/s44147-024-00512-9.
- [15] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," *J. Parallel Distrib. Comput.*, vol. 183, p. 104766, 2024, doi: 10.1016/j.jpdc.2023.104766.
- [16] M. Otair, Areej Alhmoud, H. Jia, M. Altalhi, A. M. Hussein, and L. Abualigah, "Optimized task scheduling in cloud computing using improved multi-verse optimizer," *Cluster Comput.*, vol. 25, no. 6, pp. 4221–4232, 2022, doi: 10.1007/s10586-022-03650-y.
- [17] S.-W. Zhang, J.-S. Wang, S.-H. Zhang, Y.-X. Xing, X.-F. Sui, and Y.-H. Zhang, "Task scheduling in cloud computing systems using multi-objective honey badger algorithm with two hybrid elite frameworks and circular segmentation screening," *Artif. Intell. Rev.*, vol. 58, no. 2, p. 48, 2025, doi: 10.1007/s10462-024-11032-6.
- [18] F. S. Prity, M. H. Gazi, and K. M. A. Uddin, "A review of task scheduling in cloud computing based on nature-inspired optimization algorithm," *Cluster Comput.*, vol. 26, no. 5, pp. 3037–3067, 2023, doi: 10.1007/s10586-023-04090-y.
- [19] M. Tanha, Mirsaeid Hosseini Shirvani, and A. M. Rahmani, "A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments," *Neural Comput. Appl.*, vol. 33, no. 24, pp. 16951–1698, 2021, doi: 10.1007/s00521-021-06289-9.
- [20] F. Kaplan and Ahmet Babalik, "Performance analysis of cloud computing task scheduling using metaheuristic algorithms in DDoS and normal environments," *Electronics*, vol. 14, no. 10, p. 1988, 2025, doi: 10.3390/electronics14101988.
- [21] D. Bodra, S. Khairnar, and Sushil Khairnar, "Machine learning-based cloud resource allocation algorithms: a comprehensive comparative review," *Front. Comput. Sci.*, vol. 7, p. 1678976, 2025, doi: 10.3389/fcomp.2025.1678976.

- [22] S. Kayalvili, R. Senthilkumar, S. Yasotha, and R. S. Kamalakannan, "An Optimized Resource Allocation in Cloud Using Prediction Enabled Reinforcement Learning," *Sci. Rep.*, vol. 15, p. 36088, 2025, doi: 10.1038/s41598-025-19927-2.
- [23] Y. Sanjalawe, Salam Al-E'mari, S. Fraihat, and S. Makhadmeh, "AI-driven job scheduling in cloud computing: a comprehensive review," *Artif. Intell. Rev.*, vol. 58, no. 7, p. 197, 2025, doi: 10.1007/s10462-025-11208-8.
- [24] S. I. Mohammed and Z. T. M. Al-Ta'i, "Evaluation of Resource Allocation in Cloud Using Machine Learning," in *Proceedings Proceedings of the 13th International Conference on Applied on Applied Innovations in IT in IT (ICAIIIT)*, 2025, pp. 253–260. [Online]. Available: https://opendata.uni-halle.de/bitstream/1981185920/122400/1/1-26-ICAIIIT_2025_13%282%29.pdf
- [25] F. S. Alsubaei, A. Y. Hamed, M. R. Hassan, M. Mohery, and M. K. Elnahary, "Machine learning approach to optimal task scheduling in cloud communication," *Alexandria Eng. J.*, vol. 89, pp. 1–30, 2024, doi: 10.1016/j.aej.2024.01.040.
- [26] A. Bin Naeem, Biswaranjan Senapati, J. Rasheed, J. Baili, and O. Osman, "An intelligent job scheduling and real-time resource optimization for edge-cloud continuum in next generation networks," *Sci. Rep.*, vol. 15, p. 41534, 2025, doi: 10.1038/s41598-025-25452-z.
- [27] M. F. A. Heirati and Mohammad Khalily-Dermay, "Optimized Task Scheduling in Fog-Cloud Computing Using Hybrid Deep Learning and Metaheuristic Algorithms," *Neural Process. Lett.*, vol. 58, no. 1, 2025, doi: 10.1007/s11063-025-11819-w.
- [28] D. Cui, Zhiping Peng, K. Li, Q. Li, J. He, and X. Deng, "An novel cloud task scheduling framework using hierarchical deep reinforcement learning for cloud computing," *PLoS One*, vol. 20, no. 8, p. e0329669, 2025, doi: 10.1371/journal.pone.0329669.
- [29] S. Gurusamy and R. Selvaraj, "Resource allocation with efficient task scheduling in cloud computing using hierarchical auto-associative polynomial convolutional neural network," *Expert Syst. Appl.*, vol. 249, p. 123554, 2024, doi: 10.1016/j.eswa.2024.123554.
- [30] A. Bolufé-Röhler and Dania Tamayo-Vera, "Machine learning for enhancing metaheuristics in global optimization: A comprehensive review," *Mathematics*, vol. 13, no. 18, p. 2909, 2025, doi: 10.3390/math13182909.
- [31] S. Szénási and G. Légrádi, "Machine learning aided metaheuristics: A comprehensive review of hybrid local search methods," *Expert Syst. Appl.*, vol. 258, p. 125192, 2024, doi: 10.1016/j.eswa.2024.125192.
- [32] M. Cui and Yipeng Wang, "An effective QoS-aware hybrid optimization approach for workflow scheduling in cloud computing," *Sensors*, vol. 25, no. 15, p. 4705, 2025, doi: 10.3390/s25154705.
- [33] S. Chowdhury, Ajay Katangur, and A. Sheta, "Optimization of datacenter selection through a genetic algorithm-driven service broker policy," *J. Cloud Comput.*, vol. 13, no. 1, p. 156, 2024, doi: 10.1186/s13677-024-00719-4.