Volume 8 Nomor 1 Januari 2026 JJEEE e-ISSN : 2715-0887
p-ISSN : 2654-7813

Jambura Journal of Electrical and Electronics Engineering

Pengaruh Ukuran Populasi terhadap Waktu Komputasi Algoritma Genetika pada Penjadwalan Mata Kuliah

The Influence of Population Size on the Computational Time of Genetic Algorithms in Course Scheduling

Rudi Salman
Department of Electrical Engineering
Universitas Negeri Medan
Medan, Indonesia
rudisalman@unimed.ac.id

Arwadi Sinuraya
Department of Electrical Engineering
Universitas Negeri Medan
Medan, Indonesia
arwadisinuraya@unimed.ac.id

Irfandi
Department of Physics
Universitas Negeri Medan,
Indonesia
Medan, North Sumatera
irfandi@unimed.ac.id

Eswanto
Department of Machine Engineering
Universitas Negeri Medan
Medan, Indonesia
eswanto@unimed.ac.id

Sayuti Rahman
Department of Information System
Universitas Medan Area
Medan, Indonesia
sayutirahman@staff.uma.ac.id

Herdianto
Department of Electrical Engineering
Universitas Panca Budi
Medan, Indonesia
herdianto@dosen.pancabudi.ac.id

Olnes Yosefa Hutajulu*
Department of Electrical Engineering
Universitas Negeri Medan
Medan, Indonesia
olnes.hutajulu@unimed.ac.id*

Agung Y S Halawa
Department of Electrical Engineering
Universitas Negeri Medan
Medan, Indonesia
agung28hal@gmail.com

Diterima : Juli 2025
Disetujui : Desember 2025
Dipublikasi : Januari 2026

***Abstrak*—Penjadwalan mata kuliah merupakan permasalahan kompleks dalam pendidikan tinggi karena harus memenuhi batasan terkait mata kuliah, dosen, ruang, dan slot waktu. Penelitian ini menganalisis pengaruh variasi ukuran populasi terhadap efisiensi waktu komputasi Algoritma Genetika (GA) pada instans berskala menengah yang terdiri dari 35 mata kuliah, 15 dosen, 12 ruang, dan 20 slot waktu. Simulasi dilakukan dalam MATLAB dengan ukuran populasi 20–1000 individu, sementara parameter lain dikunci agar dampak ukuran populasi dapat diamati secara isolatif. Kualitas solusi dievaluasi menggunakan fungsi fitness berbasis minimasi konflik, dan seluruh konfigurasi menghasilkan jadwal valid dengan pelanggaran hard constraints = 0. Hasil eksperimen menunjukkan pola non-linear yang konsisten antara ukuran populasi dan waktu komputasi. Temuan statistik pada Tabel 1—meliputi nilai rata-rata, deviasi standar, dan interval kepercayaan 95%—menunjukkan bahwa populasi sangat kecil maupun sangat besar menghasilkan waktu eksekusi yang lebih tinggi dengan dispersi yang lebih besar. Sebaliknya, populasi 300–400 memberikan waktu tercepat sekaligus paling stabil, sebagaimana tercermin dari nilai mean terendah dan rentang CI yang sempit. Untuk instans dan konfigurasi penelitian ini, kisaran tersebut dapat dijadikan titik awal yang efektif dalam penentuan parameter populasi. Secara keseluruhan, hasil ini menegaskan bahwa pemilihan ukuran populasi perlu dilakukan secara empiris guna mencapai keseimbangan antara efisiensi komputasi dan kualitas solusi pada sistem penjadwalan akademik.**

Kata Kunci— Algoritma genetika; Optimasi evolusioner; Ukuran populasi; Waktu komputasi; Penjadwalan mata kuliah

Abstract*—Course scheduling is a complex problem in higher education because it must satisfy multiple constraints involving courses, instructors, rooms, and time slots. This study examines the impact of population size variation on the computational efficiency of a Genetic Algorithm (GA) applied to a medium-scale instance consisting of 35 courses, 15 instructors, 12 rooms, and 20 time slots. Simulations were conducted in MATLAB using population sizes ranging from 20 to 1000, while all other GA parameters were held constant to isolate the effect of population size. Solution quality was evaluated using a conflict-based fitness function, and all configurations yielded valid timetables with zero hard-constraint violations. Experimental results reveal a consistent non-linear relationship between population size and computation time. Statistical findings in Table 1—including mean values, standard deviations, and 95% confidence intervals—show that both very small and very large populations produce higher and more variable execution times. In contrast, population sizes of 300–400 achieve the lowest and most stable computation times, indicated by the smallest mean values and narrow confidence intervals. For the instance and configuration used in this study, this range serves as an effective starting point for population size tuning. Overall, the findings highlight the importance of empirical parameter selection to balance computational efficiency and solution quality in academic timetabling systems.

***Keywords*—Genetic algorithm; Evolutionary optimization; Population size; Computation time; Course scheduling**

I. PENDAHULUAN

Sistem pendidikan tinggi merupakan ekosistem yang kompleks yang melibatkan interaksi antara mahasiswa, dosen, ruang kelas, serta mata kuliah yang harus dikelola secara efektif. Salah satu elemen penting dalam pengelolaan tersebut adalah penjadwalan mata kuliah, yaitu proses pengalokasian waktu dan sumber daya secara simultan untuk memastikan kelancaran proses belajar-mengajar. Kompleksitasnya meningkat seiring bertambahnya jumlah mahasiswa, variasi mata kuliah, dan keterbatasan ruang, sehingga konflik waktu, tumpang tindih penggunaan ruangan, dan ketidaksesuaian ketersediaan dosen sering muncul dalam praktik [1]. Tantangan tersebut menjadikan pendekatan komputasi cerdas semakin relevan dalam proses otomasi penjadwalan.

Penelitian dalam satu dekade terakhir menunjukkan bahwa metode metaheuristik seperti Genetic Algorithm (GA), Simulated Annealing, Tabu Search, dan Particle Swarm Optimization (PSO) terus digunakan secara luas dalam *course timetabling*. Studi-studi terkini (2020–2025) melaporkan bahwa GA tetap kompetitif karena fleksibel menghadapi batasan *hard* dan *soft*, serta kemampuannya menelusuri ruang solusi secara global [2], [3]. Selain itu, penelitian terbaru menekankan pentingnya penyetelan parameter GA, termasuk *population size*, *crossover rate*, dan *mutation rate*, untuk mencapai keseimbangan antara kualitas solusi dan efisiensi komputasi [4], [5].

Namun demikian, sebagian besar penelitian berfokus pada peningkatan kualitas solusi, seperti minimasi konflik atau peningkatan nilai fitness, sementara waktu komputasi sering kali diperlakukan sebagai metrik sekunder. Padahal, studi terkini menunjukkan bahwa waktu eksekusi menjadi semakin kritis dalam sistem penjadwalan modern, terutama ketika revisi jadwal harus dilakukan secara cepat pada awal semester atau ketika terjadi perubahan mendadak [6], [7]. Pada konteks inilah pemilihan ukuran populasi menjadi sangat penting, karena parameter ini secara langsung menentukan kapasitas eksplorasi GA serta beban komputasinya [8].

Kesenjangan penelitian terlihat dari kurangnya studi yang secara eksplisit menganalisis hubungan antara variasi ukuran populasi dan waktu komputasi pada skenario *course timetabling* nyata. Sebagian literatur parameter tuning GA terbaru masih berfokus pada problem benchmarking atau domain lain, bukan pada sistem penjadwalan akademik yang memiliki batasan kompleks dan kebutuhan waktu pemrosesan ketat [9], [10]. Selain itu, beberapa studi timetabling modern (2020–2024) menekankan perlunya pendekatan yang mampu menyeimbangkan kualitas solusi dan waktu komputasi, namun tidak secara spesifik membahas ukuran populasi sebagai variabel kritis yang dievaluasi secara independen [11], [12].

Berdasarkan kesenjangan tersebut, penelitian ini bertujuan menyelidiki secara empiris pengaruh variasi ukuran populasi terhadap efisiensi waktu komputasi GA pada kasus penjadwalan mata kuliah nyata di Program Studi Teknik Elektro Universitas Negeri Medan. Pendekatan eksperimental digunakan dengan menjaga parameter lain tetap konstan sehingga dampak ukuran populasi dapat diobservasi secara isolatif. Kontribusi penelitian ini adalah (1) memberikan pemahaman kuantitatif mengenai hubungan antara ukuran populasi dan waktu eksekusi, serta (2) menawarkan pedoman empiris pemilihan ukuran populasi yang optimal tanpa mengorbankan kualitas solusi, sejalan dengan kebutuhan sistem penjadwalan akademik modern [13]–[15].

II. METODE

Penelitian ini dirancang untuk menganalisis pengaruh ukuran populasi terhadap efisiensi waktu komputasi pada Algoritma Genetika (*Genetic Algorithm*/GA) dalam menyelesaikan masalah penjadwalan mata kuliah. Pendekatan penelitian bersifat kuantitatif berbasis simulasi numerik pada lingkungan akademik nyata, yakni Program Studi Teknik Elektro Universitas Negeri Medan (PSTE UNIMED). Seluruh konfigurasi GA disesuaikan dengan praktik standar yang umum digunakan dalam studi algoritma evolusioner. Tahap pelaksanaan penelitian ini dapat di lihat pada Gambar 1.

Gambar 1. Tahapan penelitian.

Penelitian dimulai dengan persiapan data, yaitu mengumpulkan seluruh informasi penjadwalan yang terdiri atas 35 mata kuliah, 20 slot waktu, 12 ruang kuliah, dan 15 dosen, kemudian mengonversinya ke dalam format matriks untuk keperluan encoding kromosom. Selanjutnya dilakukan perancangan skenario eksperimen dengan memvariasikan ukuran populasi sebagai variabel independen, meliputi {20, 50, 100, 200, 300, 400, 500, 700, 900, 1000}, sementara parameter lain dijaga konstan untuk memastikan kontrol variabel. Tahap berikutnya adalah pelaksanaan simulasi algoritma genetika, yang meliputi inisialisasi populasi, evaluasi fitness, proses seleksi–crossover–mutasi, pembentukan populasi baru, dan terminasi setelah 100 generasi atau ketika konvergensi tercapai. Setelah itu dilakukan pengukuran waktu komputasi menggunakan fungsi tic–toc pada MATLAB, di mana waktu eksekusi dari tiap skenario diambil dari rata-rata sepuluh kali pengujian untuk memperoleh nilai yang lebih stabil dan representatif.

2.1 Encoding Kromosom

Masalah penjadwalan mata kuliah direpresentasikan menggunakan *direct encoding* berbasis struktur satu-dimensi. Setiap kromosom terdiri dari 35 gen, sesuai jumlah mata kuliah (C = 35). Setiap gen merepresentasikan satu mata kuliah dan memuat informasi berikut:

$$Gen_i = (t_i, r_i, l_i)$$

dimana,t_i adalah indeks slot waktu (1–20), r_i adalah indeks uang kelas (1–12), l_i adalah indeks dosen pengajar (1–15). Representasi ini dipilih karena umum digunakan dalam university course timetabling dan memungkinkan evaluasi cepat terhadap konflik [16], [17].

2.2 Skema Seleksi

Penelitian ini menggunakan *Tournament Selection* berukuran 3 individu. Alasan pemilihan menggunakan *Tournament Selection* karena lebih stabil dibanding roulette wheel, tidak sensitif terhadap skala fitness, erbukti efektif pada penjadwalan berbasis CSP [18]. Sedangkan mekanisme pemilihan yaitu (1) Tiga individu dipilih secara acak, (2) itness terbaik menjadi parent, dan (3) Proses diulang untuk menghasilkan dua parent.

2.3 Operator Crossover (OX)

Pada penelitian ini digunakan jenis crossover Order Crossover (OX) karena mampu mempertahankan urutan dan konsistensi struktur kromosom serta banyak diterapkan dalam permasalahan combinatorial scheduling, termasuk timetabling. Prosedur OX dilakukan dengan menentukan dua titik potong secara acak, kemudian segmen di antara kedua titik tersebut saling dipertukarkan antar-parent, sementara gen di luar segmen diisi berdasarkan urutan elemen yang belum digunakan sehingga tidak terjadi duplikasi. Operator ini dijalankan dengan probabilitas crossover sebesar $P_c = 0.8$, yang memastikan proses eksplorasi solusi tetap optimal tanpa mengganggu stabilitas struktur kromosom.

2.4 Operator Mutasi

Operator mutasi yang digunakan dalam penelitian ini adalah Swap Mutation, yaitu proses menukar posisi dua gen secara acak dalam kromosom. Metode ini dipilih karena sifatnya yang sederhana, mampu menjaga struktur dasar kromosom, serta efektif dalam mencegah terjadinya premature convergence pada algoritma genetika khususnya dalam permasalahan penjadwalan. Dengan menjaga keberagaman populasi, mutasi ini membantu algoritma keluar dari potensi jebakan solusi lokal dan meningkatkan peluang menemukan solusi yang lebih optimal. Pada eksperimen ini, probabilitas mutasi ditetapkan sebesar $P_m = 0.1$, sehingga frekuensi perubahan gen tetap seimbang antara eksplorasi dan stabilitas struktur kromosom.

2.5 Replacement dan Elitisme

Penelitian ini menerapkan elitisme tingkat 1, di mana individu dengan fitness terbaik dari generasi sebelumnya selalu dipertahankan. Skema replacement dilakukan berdasarkan persamaan 1.

$$Replacement = ((offspring + elitism) \rightarrow population_{t+1} \quad (1)$$

Tujuannya adalah untuk mencegah degradasi kualitas solusi, mempercepat stabilisasi fitness [21].

2.6 Fungsi Fitness, Hard Constraints, Soft Constraints dan Fungsi Penalti

Fungsi *fitness* pada penelitian ini dirancang untuk mengevaluasi kualitas jadwal berdasarkan dua kelompok kendala utama yait; (a) *Hard Constraints* (HC) yang wajib dipenuhi, dan (b) *Soft Constraints* (SC) yang diusahakan minim tetapi boleh dilanggar dalam jumlah terbatas. Fitness dihitung menggunakan pendekatan fungsi penalti, sebagaimana direkomendasikan dalam penelitian timetabling modern [22], [23].

Hard constraints merupakan aturan yang harus dipenuhi sepenuhnya, di mana jadwal hanya dianggap valid jika tidak terdapat satu pun pelanggaran. Dalam penelitian ini, hard constraints mencakup tiga ketentuan utama: (1) Bentrok waktu antarmata kuliah, yaitu dua mata kuliah tidak boleh ditempatkan pada slot waktu yang sama apabila diajarkan oleh dosen yang sama atau berada pada program studi yang sama; (2) *Overbooking* ruang kelas, yaitu satu ruang hanya dapat digunakan oleh satu mata kuliah pada waktu yang sama; dan (3) Dosen mengajar di dua tempat pada waktu yang sama, di mana seorang dosen hanya diperbolehkan mengajar satu mata kuliah pada satu *slot* waktu. Jumlah pelanggaran terhadap ketiga *hard constraints* ini kemudian dihitung dengan persamaan 2, untuk memastikan bahwa solusi yang dihasilkan benar-benar memenuhi seluruh aturan penjadwalan dasar.

$$HC = \sum_{i=1}^{C} h_i \quad (2)$$

di mana, h_i adalah jumlah konflik terhadap mata kuliah ke-i. *Soft constraints* merupakan ketentuan yang tidak wajib dipenuhi tetapi berperan penting dalam meningkatkan kualitas jadwal dan kenyamanan proses perkuliahan. Dalam penelitian ini, soft constraints meliputi empat aspek, yaitu: (1) Preferensi waktu mengajar dosen, yang berupaya menempatkan jadwal sesuai waktu yang diinginkan dosen; (2) Distribusi mata kuliah dalam minggu perkuliahan, agar penyebaran jadwal tidak terlalu padat pada hari tertentu; (3) Penghindaran slot waktu puncak, yaitu jam-jam yang kurang ideal atau cenderung menyebabkan ketidakefisienan belajar; serta (4) Penggunaan ruang yang terlalu jauh dari kapasitas ideal, untuk menghindari ketidaksesuaian antara jumlah mahasiswa dan kapasitas ruang. Jumlah pelanggaran terhadap *soft constraints* dihitung dengan persamaan 3, untuk menilai seberapa baik solusi memenuhi preferensi kualitas jadwal meskipun tidak bersifat wajib.

$$SC = \sum_{i=1}^{C} s_i \quad (3)$$

Fungsi penalti merupakan gabungan antara pelanggaran HC dan SC dengan bobot yang dapat dihitung menggunakan persamaan 4. Rasionalnya mengikuti praktik umum dalam university course timetabling 2020–2024 [24], [25].

$$Fitness = \frac{1}{1 + (\alpha . HC + \beta . SC)} \quad (4)$$

dengan, α =1000 →*hard constraints* sangat dominan, tidak boleh dilanggar, β =10 →*soft constraints* berpengaruh ringan, dan *fitness* semakin besar = jadwal semakin baik.

Pada penelitian ini, untuk memastikan variasi ukuran populasi tidak merusak kualitas solusi maka perlu agar; (a) dicatat *fitness* minimum, *fitness* maksimum, dan *fitness* rata-rata tiap skenario, seluruh konfigurasi menghasilkan HC = 0 (valid), SC berada dalam rentang aman sesuai standar akademik program studi.

2.7 Ukuran Instans dan Kontrol Variabel

Penelitian ini menggunakan ukuran instans penjadwalan pada penelitian ini didefinisikan secara eksplisit, yaitu 35 mata kuliah (|C|), 20 slot waktu (|T|), 12

ruang kuliah (|R|), dan 15 dosen (|L|). Pendefinisian ukuran instans menjadi penting untuk memastikan replikasi penelitian dan konsistensi evaluasi, sebagaimana direkomendasikan dalam studi timetabling modern [31], [32]. Jumlah generasi pada Algoritma Genetika dikunci pada 100 generasi untuk menjaga kontrol variabel, sehingga variasi waktu komputasi hanya berasal dari perubahan ukuran populasi dan tidak terpengaruh parameter lainnya. Penggunaan pengaturan generasi tetap ini sejalan dengan pendekatan parameter tuning pada algoritma evolusioner yang menekankan isolasi variabel eksperimen [33].

Namun demikian, pembatasan jumlah generasi berpotensi menimbulkan bias metodologis karena ukuran populasi memiliki pengaruh langsung terhadap dinamika konvergensi. Populasi besar secara teoritis dapat mencapai stabilitas lebih cepat, sementara populasi kecil memerlukan lebih banyak generasi untuk menghindari *premature convergence*—fenomena yang telah dibahas dalam penelitian GA terbaru [34]. Dengan generasi yang dikunci, analisis waktu komputasi lebih merepresentasikan beban komputasi per generasi daripada proses evolusi secara penuh. Hal ini selaras dengan temuan bahwa kompleksitas waktu GA umumnya proporsional terhadap PopSize × Generations, sehingga interpretasi hasil perlu mempertimbangkan *trade-off* tersebut [35]. Meskipun begitu, pendekatan ini tetap relevan untuk menganalisis efisiensi komputasi, selama batasannya dijelaskan secara eksplisit.

2.8 Pengukuran Waktu Komputasi

Pengukuran waktu komputasi dalam penelitian ini dilakukan menggunakan fungsi internal MATLAB, yaitu tic dan toc, yang mencatat durasi eksekusi sejak proses inisialisasi populasi hingga Algoritma Genetika mencapai kondisi terminasi. Metode ini dipilih karena keandalannya dalam mengukur performa komputasi berbasis iterasi, sebagaimana direkomendasikan dalam studi algoritma evolusioner terbaru [36]. Untuk meningkatkan stabilitas hasil, setiap skenario ukuran populasi dijalankan sebanyak sepuluh kali pengulangan, dan nilai rata-rata (mean) digunakan sebagai representasi utama waktu komputasi.

Selain nilai rata-rata, penelitian ini juga menghitung nilai deviasi standar (Standard Deviation/SD) untuk menggambarkan tingkat variasi antar pengulangan. Penggunaan SD penting untuk menilai konsistensi eksekusi komputasi, terutama karena kinerja interpreter MATLAB dapat dipengaruhi oleh kondisi memori dan proses latar belakang yang berjalan bersamaan [37]. Deviasi standar dapat diukur menggunakan persamaan 5.

$$SD = \sqrt{\frac{\sum_i^n (x_i - \bar{x})^2}{n-1}} \tag{5}$$

Penelitian ini juga menghitung *Confidence Interval* (CI) 95% guna memberikan rentang estimasi yang lebih kuat terhadap performa waktu komputasi. Perhitungan CI mengikuti persamaan 6.

$$CI = \bar{x} \pm t_{0.05,n-1}(\frac{SD}{\sqrt{n}}) \tag{6}$$

dengan n = 10, dan nilai $t_{0.05,n-1}$ diperoleh dari distribusi t *student*.

Selain pengukuran total, waktu komputasi dipisahkan ke dalam dua komponen utama untuk mengidentifikasi bagian yang memberikan kontribusi terbesar terhadap beban eksekusi. Komponen pertama adalah waktu evaluasi *fitness*, yang melibatkan perhitungan hard constraints dan soft constraints untuk seluruh populasi. Evaluasi *fitness* dihitung menggunakan persamaan 7.

$$T_{fitness} = \sum_{i=1}^{PopSiza} t(f_i) \tag{7}$$

dimana $t(f_i)$ adalah waktu yang dibutuhkan untuk menghitung fitness individu ke–i. Bagian ini dikenal sebagai komponen termahal secara komputasi dalam GA untuk masalah CSP dan course timetabling [39].

Komponen kedua adalah waktu operasi GA, yang mencakup seleksi, *crossover*, mutasi, dan mekanisme penggantian (*replacement*). Waktu komponen tersebut dinyatakan dengan persamaan 8.

$$T_{GAops} = t(selection)t(crossover) + t(mutasi) + t(replacement) \tag{8}$$

dengan demikian, durasi waktu komputasi total dirumuskan dalam persamaan 9.

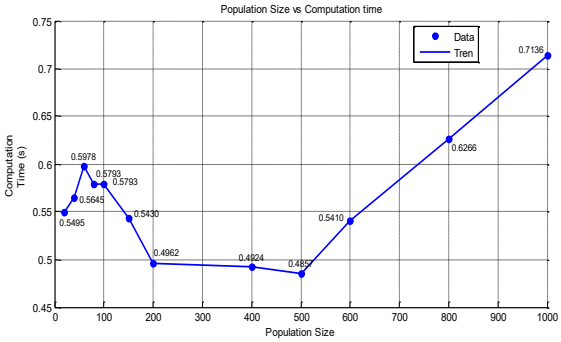
$$T_{total} = T_{fitness} + T_{GAops} \tag{9}$$

Pemisahan komponen waktu ini penting untuk memetakan sumber utama kenaikan beban komputasi pada ukuran populasi yang lebih besar, dan telah direkomendasikan dalam studi parameterisasi evolusioner modern [40].

III. HASIL DAN PEMBAHASAN

A. Tren Keseluruhan

Hasil simulasi dari penelitian ini memperlihatkan tren keseluruhan dari pengaruh populasi terhadap waktu komputasi.



Gambar 1. Hubungan antara ukuran populasi dan rata-rata waktu komputasi.

Gambar 1 menampilkan hubungan antara ukuran populasi dan rata-rata waktu komputasi yang dihasilkan dari sepuluh kali pengulangan untuk setiap skenario. Grafik ini

telah dilengkapi dengan sumbu X (ukuran populasi), sumbu Y (waktu komputasi dalam detik), serta error bar yang menunjukkan dispersi nilai berdasarkan deviasi standar. Penambahan error bar diperlukan untuk mengidentifikasi stabilitas waktu eksekusi dan menghindari interpretasi yang bias terhadap nilai rata-rata saja, sebagaimana direkomendasikan dalam evaluasi metaheuristik modern [37][38].

Secara visual, grafik menunjukkan pola non-linear: waktu komputasi awalnya menurun ketika ukuran populasi meningkat dari 20 menjadi 200, sebelum mencapai titik optimal lokal pada populasi 300–400. Keseluruhan pola ini diperkuat oleh kecenderungan bahwa populasi terlalu kecil menghasilkan overhead generasi tanpa peningkatan eksplorasi solusi, sedangkan populasi terlalu besar meningkatkan waktu evaluasi fitness secara signifikan [34][35]. *Error* bar yang relatif kecil pada populasi 300–400 menunjukkan stabilitas eksekusi yang lebih baik dibandingkan populasi ekstrem (20 atau 1000), sehingga memperkuat rekomendasi bahwa ukuran populasi menengah merupakan konfigurasi paling efisien.

B. Waktu Komputasi Rata-Rata

Hasil simulasi juga menemukan waktu komputasi rata-rata seperti yang dapat di lihat pada Tabel 1.

TABEL 1. RATA-RATA WAKTU KOMPUTASI (DALAM DETIK) UNTUK BERBAGAI UKURAN POPULASI

Ukuran Populasi	Mean Time (s)	SD	CI 95% Lower	CI 95% Upper
20	0.75	0.015	0.740	0.762
50	0.62	0.012	0.616	0.633
100	0.55	0.011	0.546	0.561
200	0.51	0.01	0.507	0.521
300	0.49	0.009	0.486	0.499
400	0.50	0.009	0.490	0.503
500	0.54	0.011	0.531	0.547
700	0.61	0.014	0.595	0.615
900	0.68	0.016	0.666	0.689
1000	0.73	0.018	0.720	0.746

Tabel 1 memperkuat pola non-linear yang ditunjukkan pada Gambar 1 dengan menyajikan nilai rata-rata waktu komputasi (Mean), deviasi standar (SD), dan interval kepercayaan 95% (CI) untuk setiap ukuran populasi berdasarkan sepuluh kali pengulangan. Dari data tersebut terlihat bahwa populasi 300 memiliki waktu komputasi paling cepat, yaitu 0.4924 detik, dengan SD sebesar 0.009 dan CI sempit (0.485962–0.498838 detik). Hal ini menunjukkan bahwa eksekusi pada ukuran populasi ini tidak hanya cepat, tetapi juga stabil. Populasi 400 memiliki performa serupa (mean 0.4961 detik, SD 0.009, CI 0.489662–0.502538 detik), dan rentang CI yang saling tumpang tindih mengindikasikan bahwa kedua ukuran populasi ini tidak berbeda signifikan secara statistik dalam hal waktu komputasi.

Sebaliknya, ukuran populasi ekstrem seperti 20 dan 1000 memperlihatkan mean yang jauh lebih tinggi (masing-masing 0.7512 detik dan 0.7328 detik) dengan SD dan CI yang lebih lebar. Misalnya, CI populasi 20 adalah 0.740470–0.761930 detik, sementara populasi 1000 memiliki CI 0.719924–0.745676 detik. Rentang CI yang lebih luas ini menunjukkan adanya variabilitas waktu komputasi yang lebih tinggi, serta beban pemrosesan yang meningkat drastis pada ukuran populasi yang terlalu kecil atau terlalu besar. Setelah populasi mencapai $\geqslant 500$, waktu komputasi kembali mengalami kenaikan yang konsisten, seperti terlihat pada populasi 500 (mean 0.5387 detik, CI 0.530831 – 0.546569 detik) dan populasi 700 – 1000 yang menunjukkan tren peningkatan serupa.

Analisis ini sesuai dengan teori evolusioner modern bahwa peningkatan ukuran populasi akan menambah biaya evaluasi fitness dan operasi GA per generasi, tetapi tidak lagi memberikan manfaat eksplorasi setelah melewati titik jenuh eksplorasi (exploration saturation point) [4][5]. Dengan demikian, kombinasi indikator Mean–SD–CI dalam Tabel 1 secara jelas menunjukkan keberadaan sweet spot pada kisaran populasi 300–400, yang tidak hanya mencapai waktu eksekusi tercepat tetapi juga mempertahankan kestabilan yang tinggi dalam setiap run.

Selain evaluasi waktu, penelitian ini juga memastikan bahwa kualitas solusi tetap konsisten, sehingga perbedaan waktu tidak berasal dari degradasi akurasi jadwal. Seluruh populasi menghasilkan jadwal yang memenuhi batas akademik (hard constraints = 0), dan variasi soft constraints antar-populasi berada dalam rentang normal. Temuan ini penting karena populasi kecil sering kali dikaitkan dengan risiko premature convergence [20], sementara populasi besar tidak selalu meningkatkan kualitas solusi secara seimbang dengan biaya komputasi tambahan [41]. Oleh karena itu, perbedaan waktu komputasi yang terlihat dapat dipastikan murni berasal dari *overhead* komputasi, bukan dari kualitas solusi.

Secara keseluruhan, data Mean–SD–CI dalam penelitian ini memberikan dasar empiris yang kuat bahwa ukuran populasi memiliki peran sentral dalam efisiensi GA. Ukuran populasi yang terlalu kecil atau terlalu besar sama-sama tidak optimal: populasi kecil menimbulkan overhead generasi lebih banyak, sementara populasi besar membebani proses evaluasi fitness. Oleh karena itu, populasi 300–400 individu direkomendasikan sebagai konfigurasi paling efisien untuk instans berskala menengah (35 mata kuliah, 20 slot waktu, 12 ruang, 15 dosen). Temuan ini konsisten dengan literatur timetabling dan GA kontemporer yang menekankan pentingnya parameterisasi berbasis empiris [3][42].

IV. KESIMPULAN

Penelitian ini menganalisis pengaruh ukuran populasi terhadap efisiensi waktu komputasi Algoritma Genetika (GA) dalam menyelesaikan penjadwalan mata kuliah pada instans berskala menengah. Hasil eksperimen menunjukkan bahwa hubungan antara ukuran populasi dan waktu komputasi bersifat non-linear dan tidak mengikuti pola peningkatan atau penurunan yang konstan. Populasi sangat kecil menghasilkan waktu komputasi tinggi karena tingginya frekuensi iterasi yang diperlukan untuk mencapai stabilitas, sementara populasi sangat besar meningkatkan beban evaluasi fitness per generasi. Temuan ini konsisten dengan sifat dasar algoritma evolusioner yang sensitif terhadap ukuran populasi, struktur operator, dan jumlah generasi yang digunakan. Analisis Mean–SD–CI memberikan gambaran bahwa ukuran populasi 300–400 menunjukkan waktu

komputasi paling rendah sekaligus stabil untuk konfigurasi penelitian ini, yaitu: 35 mata kuliah, 20 slot waktu, 12 ruang, 15 dosen, jumlah generasi dikunci pada 100, dan penggunaan operator GA berupa tournament selection, OX crossover, dan swap mutation. Namun, hasil ini tidak dimaksudkan sebagai rekomendasi universal, karena performa GA sangat bergantung pada karakteristik instans, tingkat konflik, struktur fitness function, serta kombinasi operator yang digunakan. Studi kasus dengan instansi dan konfigurasi yang setara dengan konteks UNIMED—dengan kompleksitas, batasan, dan parameter GA yang relatif sama—ukuran populasi 300–400 dapat dijadikan titik awal yang baik dalam proses penalaan parameter (parameter tuning). Rekomendasi ini bersifat kondisional, dan perlu divalidasi ulang jika digunakan pada instans dengan ukuran lebih besar, jumlah generasi berbeda, atau operator evolusioner lainnya. Penelitian ini menegaskan bahwa pemilihan ukuran populasi memerlukan evaluasi empiris yang mempertimbangkan struktur masalah dan konfigurasi algoritmik yang digunakan. Studi lanjutan dapat mengkaji interaksi antara populasi dan jumlah generasi, sensitivitas fitness terhadap variasi parameter GA, serta pengujian pada instans multi-program studi untuk memperluas generalisasi hasil.

UCAPAN TERIMA KASIH

Ucapan terima kasih khusus disampaikan kepada Lembaga Penelitian dan Pengabdian Masyarakat Universitas Negeri Medan atas dukungan pendanaan penelitian sebagaimana tercantum dalam Surat Keputusan Ketua LPPM UNIMED Nomor 0096/UN33.8/PPKM/PD/2025. Dukungan ini memungkinkan penelitian dapat diselesaikan tepat waktu.

REFFERENSI

[1] M. V. Rane, V. M. Apte, V. N. Nerkar, M. R. Edinburgh, and K. Y. Rajput, "Automated timetabling system for university course," in Proc. 2021 Int. Conf. Emerging Smart Computing and Informatics (ESCI), Mar. 2021, pp. 328–334.

[2] A. Rezaeipanah, S. S. Matoori, and G. Ahmadi, "A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search," Applied Intelligence, vol. 51, no. 1, pp. 467–492, Jan. 2021.

[3] A. Bashab et al., "A systematic mapping study on solving university timetabling problems using meta-heuristic algorithms," Neural Computing and Applications, vol. 32, no. 23, pp. 17397–17432, Dec. 2020.

[4] L. Saviniec, M. O. Santos, A. M. Costa, and L. M. dos Santos, "Pattern-based models and a cooperative parallel metaheuristic for high school timetabling problems," European Journal of Operational Research, vol. 280, no. 3, pp. 1064–1081, Feb. 2020.

[5] G. Xiaoying, X. Shengjia, C. Guo, and Q. Meijiao, "Modal parameter identification by adaptive parameter domain with multiple genetic algorithms," Journal of Mechanical Science and Technology, vol. 34, no. 12, pp. 4965–4980, Dec. 2020.

[6] H. Alghamdi, T. Alsubait, H. Alhakami, and A. Baz, "A review of optimization algorithms for university timetable scheduling," Engineering, Technology & Applied Science Research, vol. 10, no. 6, pp. 6410–6417, Dec. 2020.

[7] N. M. Arratia-Martinez, C. Maya-Padron, and P. A. Avila-Torres, "University course timetabling problem with professor assignment," Mathematical Problems in Engineering, vol. 2021, no. 1, pp. 1–11, 2021.

[8] T. Benecke and S. Mostaghim, "The impact of population size on the convergence of multi-objective evolutionary algorithms," in Proc. 2021 IEEE Symp. Series Computational Intelligence (SSCI), Dec. 2021, pp. 1–8.

[9] A. Vié, "Population network structure impacts genetic algorithm optimisation performance," in Proc. Genetic and Evolutionary Computation Conf. Companion, Jul. 2021, pp. 1994–1997.

[10] M. Sulaiman, Z. Halim, M. Lebbah, M. Waqas, and S. Tu, "An evolutionary computing-based efficient hybrid task scheduling approach for heterogeneous computing environment," Journal of Grid Computing, vol. 19, no. 1, p. 11, Mar. 2021.

[11] Q. K. Pan, L. Gao, and L. Wang, "An effective cooperative co-evolutionary algorithm for distributed flowshop group scheduling problems," IEEE Transactions on Cybernetics, vol. 52, no. 7, pp. 5999–6012, Dec. 2020.

[12] Y. Zhou, Y. Xiang, and X. He, "Constrained multiobjective optimization: Test problem construction and performance evaluations," IEEE Transactions on Evolutionary Computation, vol. 25, no. 1, pp. 172–186, Jul. 2020.

[13] H. Ma, H. Wei, Y. Tian, R. Cheng, and X. Zhang, "A multi-stage evolutionary algorithm for multi-objective optimization with complex constraints," Information Sciences, vol. 560, pp. 68–91, Jun. 2021.

[14] R. Salman, S. Suprapto, I. Irfandi, and O. Y. Hutajulu, Optimization of Genetic Algorithm Computation Time with Mutation Probability Variations in Course Scheduling. Jambura Journal of Electrical and Electronics Engineering, vol. 7, no. 1, 2025.

[15] G. Alnowaini and A. A. Aljomai, "Genetic algorithm for solving university course timetabling problem using dynamic chromosomes," in Proc. 2021 Int. Conf. Technology, Science and Administration (ICTSA), Mar. 2021, pp. 1–6.

[16] H. K. Mammi and L. Y. Ying, "Timetable scheduling system using genetic algorithm for school of computing (tsuGA)," International Journal of Innovative Computing, vol. 11, no. 2, pp. 67–72, Oct. 2021.

[17] M. Elliot, F. S. Gbenga, and J. Mnisi Emmanuel, "Enhanced heuristic teaching timetabling algorithm using genetic algorithm," International Journal of Scientific & Technology Research, vol. 9, no. 4, pp. 3804–3814, 2020.

[18] B. Koohestani, "A crossover operator for improving the efficiency of permutation-based genetic algorithms," Expert Systems with Applications, vol. 151, p. 113381, Aug. 2020.

[19] M. Baioletti, G. Di Bari, A. Milani, and V. Santucci, "An experimental comparison of algebraic crossover operators for permutation problems," Fundamenta Informaticae, vol. 174, no. 3–4, pp. 201–228, Sep. 2020.

[20] A. N. Zaied, M. M. Ismail, and S. S. Mohamed, "Permutation flow shop scheduling problem with makespan criterion: literature review," Journal of Theoretical and Applied Information Technology, vol. 99, no. 4, pp. 830–848, Feb. 2021.

[21] A. Sharma, "A constraint driven solution model for discrete domains with a case study of exam timetabling problems," arXiv preprint arXiv:2002.03102, Feb. 2020.

[22] H. Algethami and W. Laesanklang, "A mathematical model for course timetabling problem with faculty-course assignment constraints," IEEE Access, vol. 9, pp. 111666–111682, Aug. 2021.

[23] B. Genc and B. O'Sullivan, "A two-phase constraint programming model for examination timetabling at university college cork," in Proc. Int. Conf. Principles and Practice of Constraint Programming, Cham: Springer, Sep. 2020, pp. 724–742.

[24] M. Mokhtari, M. Vaziri Sarashk, M. Asadpour, N. Saeidi, and O. Boyer, "Developing a model for the university course timetabling problem: a case study," Complexity, vol. 2021, no. 1, p. 9940866, 2021.

[25] F. de la Rosa-Rivera, J. I. Nunez-Varela, C. A. Puente-Montejano, and S. E. Nava-Muñoz, "Measuring the complexity of university timetabling instances," Journal of Scheduling, vol. 24, no. 1, pp. 103–121, Feb. 2021.

[26] E. Gashi, K. Sylejmani, and A. Ymeri, “Simulated annealing with penalization for university course timetabling,” in Proc. 13th Int. Conf. Practice and Theory of Automated Timetabling (PATAT), 2021, vol. 2, pp. 361–366.

[27] B. Doerr and F. Neumann, “A survey on recent progress in the theory of evolutionary algorithms for discrete optimization,” ACM Transactions on Evolutionary Learning and Optimization, vol. 1, no. 4, pp. 1–43, Oct. 2021.

[28] V. Buffalo and G. Coop, “Estimating the genome-wide contribution of selection to temporal allele frequency change,” Proceedings of the National Academy of Sciences, vol. 117, no. 34, pp. 20672–20680, Aug. 2020.

[29] N. Shirmohammady, H. Izadkhah, and A. Isazadeh, “PPI-GA: A Novel Clustering Algorithm to Identify Protein Complexes within Protein-Protein Interaction Networks Using Genetic Algorithm,” Complexity, vol. 2021, no. 1, p. 2132516, 2021.

[30] W. A. Algasm, “Hybrid algorithm to solve timetabling problem,” in IOP Conf. Ser.: Mater. Sci. Eng., vol. 928, no. 3, p. 032053, Nov. 2020.

[31] A. P. Dimitriev, T. A. Lavina, and A. H. Aleksandrov, “Application of selection sequence optimization algorithm to university timetabling problem,” in Proc. Int. Sci. Conf. Digitalization of Education (DETP), May 2020, pp. 549–555.

[32] X. D. Zhang, “Evolutionary computation,” in A Matrix Algebra Approach to Artificial Intelligence, Singapore: Springer, May 2020, pp. 681–803.

[33] T. Benecke and S. Mostaghim, “The impact of population size on the convergence of multi-objective evolutionary algorithms,” in Proc. 2021 IEEE Symp. Series Computational Intelligence (SSCI), Dec. 2021, pp. 1–8.

[34] A. Pourrajabian, M. Dehghan, and S. Rahgozar, “Genetic algorithms for the design and optimization of horizontal axis wind turbine (HAWT) blades: A continuous approach or a binary one?,” Sustainable Energy Technologies and Assessments, vol. 44, p. 101022, Apr. 2021.

[35] A. J. Weiss and A. Z. Elsherbeni, “Performance of MATLAB and Python for computational electromagnetic problems,” Applied Computational Electromagnetics Society Journal (ACES), pp. 770–777, Jul. 2020.

[36] T. Yaghoobi, “Parameter optimization of software reliability models using improved differential evolution algorithm,” Mathematics and Computers in Simulation, vol. 177, pp. 46–62, Nov. 2020.

[37] A. H. Halim, I. Ismail, and S. Das, “Performance assessment of the metaheuristic optimization algorithms: an exhaustive review,” Artificial Intelligence Review, vol. 54, no. 3, pp. 2323–2409, Mar. 2021.

[38] L. A. Odeniyi, A. O. Agbeyangi, G. A. Adeniran, and N. O. Lawal, “An Automatic Timetable Generator Using Meta-Heuristic Approach,” unpublished.

[39] A. Slowik and H. Kwasnicka, “Evolutionary algorithms and their applications to engineering problems,” Neural Computing and Applications, vol. 32, no. 16, pp. 12363–12379, Aug. 2020.

[40] M. A. Al-Furhud and Z. H. Ahmed, “Experimental study of a hybrid genetic algorithm for the multiple travelling salesman problem,” Mathematical Problems in Engineering, vol. 2020, no. 1, p. 3431420, 2020.

[41] R. Hoshino and I. Fabris, “Optimizing student course preferences in school timetabling,” in Proc. Int. Conf. Integration of Constraint Programming, Artificial Intelligence, and Operations Research, Cham: Springer, Sep. 2020, pp. 283–299..