OPTIMISASI PENJADWALAN MATA KULIAH MENGGUNAKAN ALGORITMA *SIMULATED ANNEALING*

Andre Yonathan Sukhoco[1], Ferdinand Lanvino, Ekabrata Yudhistyra, Budi Permana, Kurweni Ukar

STMIK LIKMI

[1]andre@likmi.ac.id

ABSTRAK

Penyusunan jadwal perkuliahan merupakan aktivitas administratif yang rutin dilakukan setiap pergantian semester di lingkungan institusi pendidikan tinggi, salah satunya di STMIK LIKMI. Namun, kompleksitas kombinasi data menghambat pihak administrasi untuk bisa menuntaskan aktivitas tersebut secara cepat. Salah satu alternatif yang dapat membantu proses penjadwalan adalah dengan menerapkan algoritma optimisasi, yaitu *Simulated Annealing* (SA).

Proses penjadwalan dilakukan secara iteratif, dimulai dengan membentuk solusi jadwal baru berdasarkan solusi saat ini, mengevaluasi pelanggaran batasan-batasan yang telah ditentukan terhadap solusi baru tersebut, dan memutuskan apakah menerima solusi baru tersebut untuk dianilkan lebih lanjut atau tidak berdasarkan probabilitas dalam setiap iterasi, layaknya proses anil yang asli. Proses tersebut berlangsung sampai mencapai batas tertentu dan menghasilkan solusi jadwal yang paling memungkinkan yang bisa diperoleh.

Proses penjadwalan yang menerapkan algoritma SA tersebut dapat disesuaikan secara fleksibel dengan kebutuhan studi kasus, sehingga dapat menghasilkan solusi jadwal terbaik yang beragam dan dapat memenuhi batasan-batasan yang ditentukan semaksimal mungkin, dengan mengompromikan kecepatan proses dan keidealan solusi.

Kata kunci: Penjadwalan, Simulated Annealing, Metaheuristik, Optimisasi

ABSTRACT

Timetabling is an administrative activity that is routinely carried out every semester in higher education institutions, for example at STMIK LIKMI. However, the complexity of data combination prevents the administration from being able to complete these activities quickly. One alternative that can help the scheduling process is to apply an optimization algorithm, namely Simulated Annealing (SA).

The timetabling process is carried out iteratively, starting by forming a new schedule solution based on the current solution, evaluating the violation of predetermined constraints on the new solution, and deciding whether to accept the new solution for further annealing or not based on the probability in each iteration, as the original annealing process. This process continues until it reaches a certain limit and produces the most possible schedule solution that can be obtained.

The scheduling process that applies the SA algorithm can be adjusted flexibly to the needs of the case study, so that it can produce a variety of best schedule solutions and can meet specified constraints as much as possible, by compromising process speed and solution ideality.

Keywords: Timetabling, Simulated Annealing, Metaheuristics, Optimization

1. PENDAHULUAN

Penyusunan jadwal di lingkungan institusi pendidikan tinggi dilakukan setiap menjelang dimulainya semester perkuliahan. Pada masa waktu ini, pihak administrasi kampus melakukan pengolahan data-data pendaftaran perkuliahan dari mahasiswa dan ketersediaan mengajar dosen untuk menentukan jadwal perkuliahan. Proses penjadwalan ini dilakukan sedemikian rupa supaya baik dari sisi mahasiswa dan dosen masing-masing tidak mendapatkan jadwal lebih pada waktu yang bersamaan. Pasalnya, proses penjadwalan ini mendatangkan kesulitan tersendiri karena banyaknya kombinasi dari berbagai dimensi yang harus diuji demi mendapatkan solusi jadwal terbaik. Di sisi lain, kesempatan waktu yang tersedia untuk melakukan penyusunan jadwal perkuliahan hanya terbatas, yaitu waktu setelah registrasi semester baru hingga menjelang dimulainya perkuliahan semester baru.

Dari situasi yang dihadapkan, administrasi kampus membutuhkan percepatan dalam penyusunan jadwal kuliah, sehingga masa waktu yang ada bisa dimanfaatkan untuk aktivitas penting lainnya. Percepatan ini bisa dilakukan dengan memanfaatkan sumber daya komputasi dan algoritma yang sesuai untuk mendapatkan solusi susunan jadwal terbaik yang memenuhi kondisi-kondisi yang ditentukan. Proses penjadwalan itu sendiri merupakan salah satu permasalahan optimisasi kombinatorial yang rumit dan sulit dipecahkan dengan menggunakan teknik konvensional. Beberapa jenis metodologi dapat digunakan untuk memecahkan masalah penjadwalan seperti *operational research*, *single solution-based meta-heuristic*, *population-based meta-heuristic*, *hyper-heuristic*, *multi criteria*/*objective*, dan *hybrid approaches*. Metodologi *single solution-based meta-heuristic* atau dikenal juga sebagai algoritma *local search* dinilai lebih efisien karena mudah diimplementasikan dan mampu menghasilkan solusi berkualitas tinggi dengan memakai daya komputasi yang wajar [1]. Salah satu algoritma local search yang dapat digunakan adalah algoritma optimisasi *simulated annealing*.

Penelitian sebelumnya [2] mencoba memecahkan masalah penjadwalan ujian pada dataset Carter dengan menggabungkan *algoritma greedy* dan *simulated annealing*. Algoritma *greedy* digunakan untuk menghasilkan solusi awal penjadwalan yang kemudian akan dioptimisasi dengan menggunakan *simulated annealing*. Hasilnya, *simulated annealing* terbukti mengungguli algoritma *multi-start hill climbing* dan *late acceptance hyper-heuristics*. Penelitian lainnya [3] dilakukan dengan melakukan perbandingan implementasi antara *simulated annealing*, *genetic algorithm*, dan *simulated annealing-genetic algorithm hybrid* terhadap dataset KTH Royal Institute of Technology. Hasilnya, *simulated annealing* secara konsisten memenuhi seluruh kriteria *fitness* dan proses komputasinya secara signifikan lebih cepat daripada *genetic algorithm*.

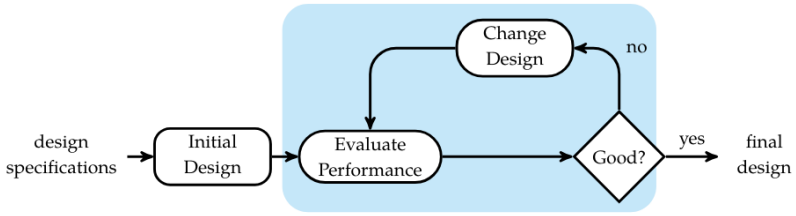
Penelitian yang dikemukakan dalam artikel ini akan menunjukkan implementasi algoritma *simulated annealing* sebagai salah satu alternatif dalam mengoptimisasi proses penjadwalan kuliah bagi lingkungan institusi pendidikan tinggi. Penelitian ini memperhatikan berbagai kondisi dan kebutuhan Sekolah Tinggi Manajemen dan Ilmu Komputer (STMIK) LIKMI sebagai subjek penelitian.

2. KAJIAN TEORI

2.1. Optimisasi

Optimisasi merupakan tindakan untuk memperoleh hasil terbaik dalam kondisi tertentu. Selain itu, optimisasi dapat didefinisikan sebagai proses menemukan kondisi yang dapat memberikan nilai minimum atau maksimum dari sebuah fungsi [4]. Optimisasi dapat menjadi penting dan berguna dalam situasi tertentu, terutama ketika menghadapi permasalahan yang sulit di mana tidak adanya model teoretis yang memadai untuk menentukan solusi [5].

Rancangan proses optimisasi dimulai dengan mendefinisikan spesifikasi masalah yang merincikan parameter, konstanta, objektif, dan batasan yang akan dicapai. Gambaran proses yang dimaksud dapat dilihat pada Gambar 1.



Gambar 1. Rancangan proses optimisasi. Area biru adalah area fokus optimisasi [6, p. 4].

2.2. Permasalahan Penjadwalan Kuliah

Penjadwalan kuliah merupakan permasalahan optimisasi hibrida dalam kelas masalah *NP-hard* (*non-deterministic polynomial*) yang terjadi setiap menuju awal semester akademik kampus, meliputi alokasi peristiwa (*events*) (mata kuliah, pengajar, dan peserta didik) ke dalam sumber daya (*resources*) slot waktu dan ruangan yang bersifat tetap. Permasalahan ini harus memenuhi syarat atau batasan keras (*hard constraints*) dan lunak (*soft constraints*) selama proses alokasi peristiwa ke dalam sumber daya, supaya jadwal-jadwal yang memungkinkan dapat diperoleh setelah semua syarat terpenuhi dan menaikkan kualitas jadwal yang dihasilkan selama dibutuhkan [7, p. 43].

Kompleksitas yang muncul di dalam permasalahan penjadwalan kuliah di antaranya:

a. Proses penjadwalan itu sendiri awalnya merupakan kelas masalah *NP-complete*, kemudian tidak dapat diselesaikan dalam kelas waktu polinomial karena pertumbuhan masalah secara eksponensial dan adanya variasi dalam pertumbuhan jumlah mahasiswa. Untuk itu, permasalahan ini memerlukan penyelesaian dengan pendekatan heuristik.
b. Jumlah batasan, baik keras maupun lunak, berbeda dari satu institusi dengan institusi lainnya. Untuk itu, tujuan utama dari algoritma adalah untuk memaksimalkan jumlah batasan lunak yang terpenuhi di dalam jadwal akhir.

Menurut [7, p. 44], berikut ini adalah definisi-definisi terkait permasalahan penjadwalan kuliah:

a. *Event* merupakan aktivitas yang dijadwalkan, meliputi pengajar, mata kuliah, dan peserta didik;
b. *Timeslot*, merupakan interval waktu daripada *event* yang dijadwalkan;
c. *Resource*, merupakan sesuatu yang digunakan oleh *event*, seperti peralatan, ruangan, slot waktu, dan lain-lain;
d. *Constraint*, merupakan batasan dalam menjadwalkan *event*, terbagi menjadi dua jenis yaitu batasan keras dan batasan lunak. Batasan keras harus terpenuhi seutuhnya supaya solusi yang dihasilkan menjadi mungkin dan tanpa konflik, sedangkan batasan lunak terkait dengan fungsi objektif, yaitu memaksimalkan jumlah batasan yang terpenuhi. Batasan lunak tidak perlu terpenuhi seutuhnya, tetapi semakin banyak jumlah batasan lunak yang terpenuhi akan membuat kualitas solusi terhitung dalam fungsi objektif meningkat;
e. *People*, merupakan individu terkait *event*, seperti dosen dan mahasiswa;
f. *Conflict*, merupakan kondisi bentrok antara dua *event*, seperti jadwal banyak dosen dalam satu ruangan dan waktu yang sama.

Berdasarkan ulasan yang dilakukan oleh [7] terhadap ragam pendekatan untuk menyelesaikan permasalahan penjadwalan kuliah, satu hal yang disorot adalah pendekatan metaheuristik yang disebut lebih efisien untuk eksplorasi ruang solusi dan disandingkan

dengan teknik baru yang cerdas untuk menganalisis jenis permasalahan tersebut. Adapun hal-hal yang perlu ditekankan adalah tentang independensi proses penjadwalan, negosiasi agen untuk bisa menghapus gangguan *event* dan *resource* satu dengan lainnya, fleksibilitas agen untuk mengombinasikan berbagai metode heuristik yang berbeda dalam menjadwalkan *event* tunggal biasa, sehingga bisa memperoleh sudut pandang yang utuh dan beralasan atas struktur algoritma yang tersedia.

2.3. Algoritma *Simulated Annealing* (SA)

Simulated annealing adalah salah satu metode metaheuristik yang paling sederhana dan terkenal untuk menangani permasalahan optimisasi global *black box* yang sulit, di mana fungsi objektif tidak diberikan secara eksplisit dan hanya bisa dievaluasi melalui simulasi komputer yang memakan biaya. Tiga ilmuwan IBM yaitu Kirkpatrick, Gelatt, dan Vecchi, memperkenalkan algoritma SA sekitar tahun 1980an dengan menganalogikan proses anil material secara fisik [8, p. 3].

Algoritma SA dikembangkan dari dua algoritma optimisasi pendahulunya, yaitu algoritma *local search* (Monte Carlo) dan Metropolis.

a. Algoritma *local search* (Monte Carlo)

   Algoritma ini merupakan algoritma iteratif yang pencariannya dimulai dari sebuah titik solusi awal i dalam area solusi S secara acak dan dinilai layak. Sebuah mekanisme generasi diterapkan secara terus menerus untuk menemukan solusi j yang lebih baik ($f(j)$), dengan cara menelusuri ketetanggaan solusi dari solusi saat ini. Jika solusi yang dimaksud ditemukan, solusi tersebut menjadi solusi saat ini. Algoritma ini berhenti ketika tidak ada lagi peningkatan solusi yang bisa ditemukan dan solusi terakhir saat ini dianggap sebagai solusi perkiraan untuk sebuah permasalahan optimisasi.

   Algoritma minimasi ini dapat dirangkum dalam *pseudocode* sebagai berikut [8, pp. 4-5]:

   1. Buat solusi awal i;
   2. Generasi solusi j dari area ketetanggaan S_i solusi i;
   3. Jika $f(j) < f(i)$ maka j menjadi solusi saat ini;
   4. Jika $f(j) \geq f(i)$ untuk semua $j \in S_i$ maka SELESAI;
   5. Kembali ke langkah 2.

   Menurut definisi yang diungkapkan oleh [8, p. 5], sebuah solusi ($i^* \in S$) dapat disebut sebagai optimum lokal jika nilai dari fungsi lokal optimum tersebut ($f(i^*)$) lebih kecil atau sama dengan nilai dari fungsi solusi yang lain ($f(j)$) dalam area ketetanggaan optimum lokal. Definisi berikutnya yang diungkapkan adalah area ketetanggaan yang ideal, di mana setiap optimum lokal yang teridentifikasi merupakan solusi terbaik (optimum global) untuk penyelesaian permasalahan.

   Kondisi ketetanggan ideal tersebut dapat mengarahkan algoritma ke solusi terbaik secara sempurna. Namun, ketetanggaan ideal biasanya tidak berguna, karena usaha untuk mencari solusi terbaik harus dilakukan dengan cara memeriksa satu per satu solusi dalam S (enumerasi penuh) yang tentu sangat memakan waktu dan berpotensi tidak mungkin dilakukan untuk menyelesaikan permasalahan yang besar.

   Artinya, jika solusi saat ini menurut nilai fungsi objektifnya sudah mencapai optimum lokal, maka algoritma pencarian akan terus terjebak dalam posisi ini sampai ia bisa menemukan struktur ketetanggaan yang dapat menghasilkan nilai fungsi

objektif "keluar" dari area optimum lokal. Untuk mencegah algoritma ini terjebak dalam situasi tersebut (minimum lokal), maka dibutuhkan definisi proses yang akan menerima transisi solusi walau secara sementara akan mendapatkan performa solusi saat ini yang lebih rendah.

b. Algoritma Metropolis
Algoritma ini dikembangkan oleh tiga ilmuwan Amerika, yaitu Metropolis, Rosenbluth, dan Teller, pada tahun 1953, yang menyimulasikan proses anil fisik dan menggunakan algoritma Monte Carlo sebagai dasarnya. Dimulai dari keadaan awal i dari energi E_i, keadaan baru j dari energi E_j dihasilkan dengan mengubah posisi salah satu partikel. Jika selisih energi, $E_i - E_j$, bernilai positif (keadaan baru berenergi lebih rendah), maka keadaan *j* menjadi keadaan baru saat ini. Namun sebaliknya, jika selisih energi lebih rendah atau sama dengan nol, maka peluang keadaan *j* menjadi keadaan saat ini ditentukan dengan Persamaan 1.

$$Pr\{\text{keadaan sekarang } j\} = e^{\left(\frac{E_i - E_j}{k_b T}\right)}$$

Persamaan 1. Probabilitas penerimaan keadaan baru [8, p. 5]

di mana T mewakili suhu benda padat dan k_b adalah konstanta Boltzmann ($k_b = 1.38 \times 10^{-23}$ joule/Kelvin).
Kriteria penerimaan keadaan baru disebut dengan kriteria Metropolis. Jika pendinginan dilakukan secara perlahan, benda padat mencapai keadaan ekuilibrium pada setiap suhu T yang diberikan. Dalam algoritma Metropolis, ekuilibrium diperoleh dengan menghasilkan banyak transisi di setiap suhu [8, p. 6].

Dalam algoritma SA seperti yang diungkapkan oleh [8, p. 7], algoritma Metropolis diterapkan untuk menghasilkan sebuah urutan solusi dalam ruang keadaan S. Untuk melakukannya, sebuah analogi dibuat antara sistem multipartikel dan permasalahan optimisasi dengan menggunakan kesetaraan berikut:

a. Titik-titik keadaan-ruang mewakili kemungkinan keadaan benda padat;
b. Fungsi yang hendak diminimasi mewakili energi benda padat.

Dengan dasar analogi tersebut, maka prinsip penerimaan suatu solusi, dengan *c* sebagai parameter kontrol, i dan j sebagai dua titik dalam ruang keadaan, didefinisikan pada Persamaan 2.

$$Pr\{\text{terima } j\} = \begin{cases} 1 & \text{jika } f(j) < f(i) \\ e^{\left(\frac{f(i) - f(j)}{c}\right)} & \text{lainnya.} \end{cases}$$

Persamaan 2. Prinsip penerimaan solusi [8, p. 6]

Selain itu, prinsip generasi sebuah tetangga selaras dengan mekanisme perturbasi dari algoritma Metropolis dan prinsip penerimaan mewakili kriteria Metropolis. Definisi lain yang diungkapkan adalah transisi solusi, yaitu penggantian solusi saat ini dengan solusi tetangga yang dilakukan dalam dua langkah: generasi dan penerimaan [8, p. 7].

Prinsip algoritma SA dapat dirangkum dalam *pseudocode* di bawah ini [8, pp. 7-8]:

1. Inisialisasi $i := i_{\text{start}}$, $k := 0$, $c_k = c_0$, $L_k := L_0$;
2. Ulangi
3. Untuk $l = 0$ sampai L_k lakukan
   a. Generasi sebuah solusi j dari ketetanggaan S_i dari solusi saat ini i;
   b. Jika $f(j) < f(i)$, maka $i := j$ (j menjadi solusi saat ini);

c. Selain itu, j menjadi solusi saat ini dengan probabilitas $e^{\left(\frac{f(i)-f(j)}{c_k}\right)}$;

4. $k := k + 1$;
5. Hitung(L_k, c_k);
6. Sampai $c_k \simeq 0$.

c_k adalah parameter suhu, L_k adalah jumlah transisi yang dibuat pada k iterasi.

Salah satu fitur utama dari algoritma SA adalah adanya kemampuan untuk menerima transisi solusi yang dapat mendegradasi fungsi objektif. Pada permulaan proses, nilai suhu c_k terbilang tinggi, yang memungkinkan adanya transisi solusi dengan tingginya degradasi objektif, dan dengan demikian dapat menelusuri ruang keadaan secara menyeluruh. Saat c_k berkurang, hanya transisi yang meningkatkan objektif atau penurunan objektif yang rendah yang diterima. Hingga pada akhirnya, ketika c_k mendekati nol, tidak ada penurunan objektif lagi yang dapat diterima, dan algoritma SA sudah berperilaku seperti algoritma Monte Carlo.

Upaya pendinginan dalam algoritma SA dilakukan salah satunya dengan pendekatan geometris [8, pp. 15-18]. Nilai suhu awal c_0 dicari dengan tujuan memastikan tingginya tingkat penerimaan solusi pada tahap awal iterasi. Untuk itu, c_0 diawali dengan nilai yang rendah, kemudian secara progresif dikalikan dengan angka lebih besar dari satu sampai tingkat penerimaan $\chi(c_0)$ mendekati satu. Penurunan parameter suhu c_k dalam setiap iterasi ditentukan dengan $c_{k+1} := \alpha c_k$, di mana umumnya $0.8 < \alpha < 0.99$. Jika iterasi demi iterasi tidak memberikan perubahan solusi yang lebih baik dalam waktu yang lama, sebaiknya proses tersebut dihentikan. Namun, supaya algoritma bisa tetap melakukan penelusuran solusi secara memadai dan memberikan kesempatan algoritma bisa mencapai konvergensi ke solusi yang terbaik, maka perlu untuk menentukan jumlah iterasi atau transisi minimum.

3. METODOLOGI PENELITIAN

Penelitian ini menggunakan metode penelitian sebagai berikut:

a. Pengumpulan data
   Data yang digunakan dalam penelitian ini adalah data mahasiswa, dosen, ruangan, dan pendaftaran kuliah untuk Semester Genap tahun akademik 2022/2023 STMIK LIKMI dari jenjang Sarjana. Seluruh data diperoleh dalam format lembar kerja (*spreadsheet*) yang diambil dari basis data yang disediakan oleh kampus.
b. Analisis dan perancangan
   Pada tahap ini, seluruh data yang dikumpulkan kemudian dianalisis untuk menentukan batasan-batasan dan *resources* untuk penjadwalan. Data kemudian dipersiapkan untuk bisa digunakan di dalam program.
c. Implementasi
   Sebuah program penjadwalan dibuat menggunakan Jupyter Notebook dengan mengimplementasikan algoritma SA diikuti penyesuaian sesuai batasan dan kebutuhan. Hasil dari program yang dijalankan berupa visualisasi jadwal perkiraan dalam bentuk *timetable*, data-data pelanggaran batasan yang tersisa, dan metrik terkait proses penjadwalan yang sudah dijalankan.

3.1. Penentuan Batasan-Batasan

Untuk memastikan algoritma dapat berjalan sesuai dengan kebutuhan, maka perlu menentukan batasan-batasan keras dan lunak. Tabel 1 menampilkan daftar batasan-batasan keras, sedangkan Tabel 2 menunjukkan daftar batasan lunak terkait penjadwalan kuliah di STMIK LIKMI.

Tabel 1. Daftar batasan keras (*hard constraints*)

Kode	Batasan
H1	Kelas reguler dapat dijadwalkan di empat slot waktu awal (slot waktu pertama sampai keempat) di semua hari.
H2	Kelas karyawan dapat dijadwalkan di dua slot waktu terakhir (slot waktu kelima dan keenam) dan jangan dijadwalkan pada hari Sabtu.
H3	Jumlah jadwal untuk masing-masing kelas disesuaikan dengan angka satuan kredit semester (SKS) masing-masing mata kuliah. Satu SKS setara 50 menit sesi dan dalam satu sesi dibatasi paling lama 100-150 menit atau setara dua sampai tiga SKS. Artinya, jika angka SKS lebih besar dari tiga, maka kelas tersebut dapat dijadwalkan lebih dari sekali dalam seminggu secara proporsional.
H4	Jenis kuliah yang tersedia adalah teori dan praktik. Kuliah teori menggunakan ruangan teori, sedangkan kuliah praktik menggunakan ruangan praktik.
H5	Terdapat mata kuliah praktik tertentu yang hanya bisa dijadwalkan di ruangan praktik yang sudah ditetapkan karena kebutuhan sarana kuliah yang hanya tersedia di ruangan tersebut.
H6	Jumlah mahasiswa di setiap kelas tidak boleh melebihi kapasitas ruangan atau sampai toleransi kapasitas 150%.
H7	Setiap ruangan hanya boleh diisi oleh paling banyak satu jadwal kelas dalam satu slot waktu.
H8	Dosen tidak boleh mengajar lebih dari sekali dalam slot hari dan waktu yang sama.

Tabel 2. Daftar batasan lunak (*soft constraints*)

Kode	Batasan
S1	Mahasiswa tidak boleh mendapat jadwal kelas lebih dari sekali dalam slot hari dan waktu yang sama.
S2	Dosen sebaiknya mendapatkan paling banyak empat jadwal kelas dalam satu hari.

4. HASIL DAN PEMBAHASAN

4.1. *Plotting* Kelas Mata Kuliah

Proses penjadwalan dipisahkan dalam dua tahap yaitu tahap perencanaan komposisi kelas (*plotting* kelas) dan penjadwalan itu sendiri menggunakan algoritma SA. Dalam tahap perencanaan komposisi kelas, data pendaftaran mahasiswa dikalkulasi untuk menentukan jumlah kelas masing-masing mata kuliah sesuai jumlah SKS dan jumlah mahasiswa masing-masing kelas berdasarkan ketersediaan dan kapasitas ruangan.

Pada tahap ini, batasan H3 dan H6 telah dipenuhi lebih dulu dengan maksud untuk meringankan proses penjadwalan yang akan menggunakan algoritma SA. Dengan demikian, laporan hasil penjadwalan bisa diperoleh lebih cepat untuk kemudian bisa segera ditindaklanjuti, apakah akan disesuaikan secara manual atau mengulang tahapan proses penjadwalan.

4.2. Implementasi Algoritma SA

Penerapan algoritma SA dalam proses penjadwalan melibatkan sejumlah penyesuaian fungsi, yaitu:

a. Fungsi penyaringan slot yang tersedia dibuat untuk memperoleh matriks slot waktu dan hari yang tersedia berdasarkan spesifikasi kelas yang akan menempati slot tersebut. Fungsi ini akan membantu pembentukan solusi jadwal yang bisa memenuhi batasan H1, H2, H4, dan H5 secara langsung. Dengan demikian, evaluasi angka pelanggaran batasan-batasan tersebut bisa dilewati;
b. Fungsi penghasil solusi awal dibuat untuk menghasilkan solusi awal berdasarkan data kelas yang sudah dirancang, dengan menggunakan fungsi penyaringan slot;
c. Fungsi relokasi kelas dibuat untuk memindahkan sebuah kelas ke slot lain berdasarkan solusi saat ini dengan memperhatikan spesifikasi kelas yang berpindah tersebut menggunakan fungsi penyaringan slot. Hasil fungsi ini adalah solusi baru dengan perbedaan kelas yang sudah berpindah slot;
d. Fungsi evaluasi solusi dibuat untuk mengevaluasi angka pelanggaran negatif batasan H7, H8, S1, dan S2 daripada solusi saat ini dan menjalankan fungsi relokasi kelas ketika pelanggaran batasan tersebut ditemukan. Hasil dari fungsi ini adalah solusi dan angka pelanggaran terbaru untuk diproses di fungsi berikutnya. Angka pelanggaran dapat disebut juga *fitness* yang dimulai dari angka pelanggaran terendah yang berarti solusi terburuk dan ditargetkan mencapai angka nol yang berarti solusi mendekati ideal;
e. Fungsi anil adalah fungsi utama yang menerapkan algoritma SA. Fungsi evaluasi solusi digunakan dalam iterasi untuk menentukan apakah menerima solusi baru yang lebih baik secara langsung atau menerima solusi baru melalui kriteria penerimaan atas hasil perhitungan probabilitas. Pendekatan yang digunakan dalam penurunan suhu adalah pendekatan geometris. Selain itu, terdapat penyesuaian algoritma melalui mekanisme untuk mengulang proses SA dari temperatur awal ketika sudah mencapai batas jumlah iterasi yang terjebak. Hal ini dilakukan untuk memberikan kesempatan iterasi tambahan bagi proses SA melanjutkan pencarian solusi yang lebih baik. Pengulangan proses SA dikendalikan oleh parameter tambahan yang bisa diatur oleh pengguna. Fungsi anil berakhir ketika temperatur kurang dari angka satu atau batas mengulang iterasi sudah tercapai.

4.3. Hasil Penjadwalan Kuliah

Parameter untuk konfigurasi proses penjadwalan ditentukan pada Tabel 3.

Tabel 3. Parameter konfigurasi penjadwalan

Parameter	Nilai
Percobaan	10
Temperatur awal	1000
Faktor penurunan	0.99
Kriteria penerimaan	0.8
Target temperatur	1
Target fitness	0
Batas iterasi terjebak	50
Batas mengulang iterasi	3

Berikut ini adalah hasil dari tahapan proses penjadwalan kuliah yang dilakukan:

a. Penjadwalan untuk kelas Reguler saja.

Tabel 4. Hasil *plotting* kelas Reguler

Data	Jumlah
Mahasiswa	186 orang
Kelas Teori	24 kelas
Kelas Praktik	22 kelas

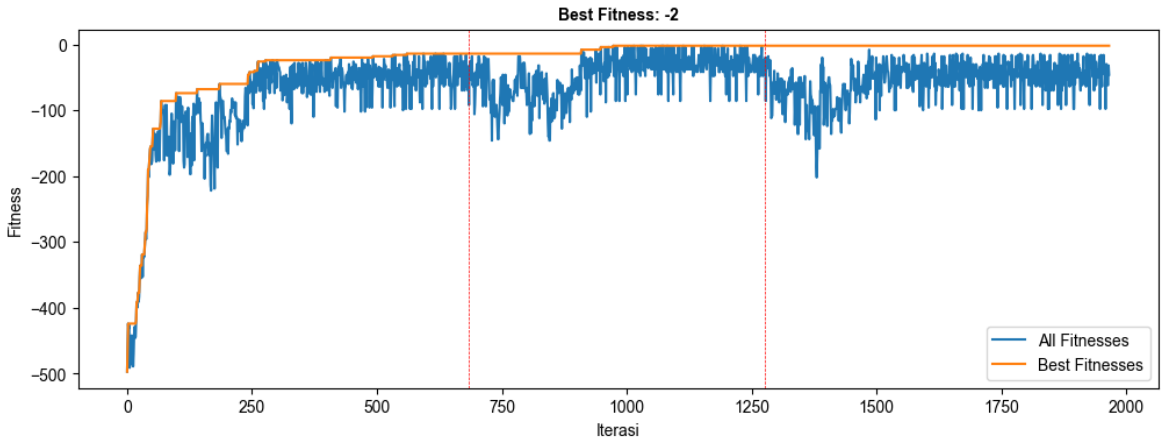
Tabel 5. Hasil percobaan penjadwalan kelas Reguler

	Percobaan										$\bar{x}$
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Fitness Awal	-586	-518	-443	-476	-433	-554	-490	-430	-497	-507	-493,4
H1	0	0	0	0	0	0	0	0	0	0	0,0
H2	0	0	0	0	0	0	0	0	0	0	0,0
H3	0	0	0	0	0	0	0	0	0	0	0,0
H4	0	0	0	0	0	0	0	0	0	0	0,0
H5	0	0	0	0	0	0	0	0	0	0	0,0
H6	0	0	0	0	0	0	0	0	0	0	0,0
H7	0	0	0	0	0	0	0	0	0	0	0,0
H8	0	0	0	0	0	0	0	0	0	0	0,0
S1	-16	-20	-26	-12	-10	-28	-14	-20	-2	-14	-16,2
S2	0	0	0	0	0	0	0	0	0	0	0,0
Fitness Akhir	-16	-20	-26	-12	-10	-28	-14	-20	**-2**	-14	-16,2
Ulang	3	0	0	1	0	0	0	3	2	0	0,9
Iterasi	2164	688	688	1335	688	688	688	2178	1965	688	1177
Durasi (s)	126,68	41,57	38,91	67,47	37,96	37,99	37,98	122,97	111,21	35,86	65,86

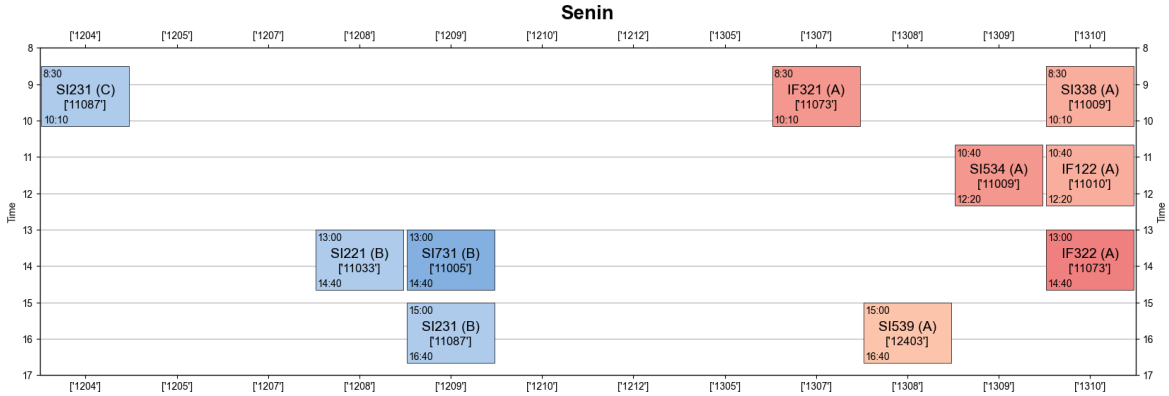
```
S1      : 2
Mahasiswa tidak boleh mendapat jadwal kelas lebih dari sekali dalam slot hari dan waktu yang sama.
```

	hari_id	timeslot	mahasiswa	counts	mata_kuliah_id	kode	nama_kuliah	sks	tipe_kuliah	shift	niks	kelas	ruang
0	3	0	2020130026	2	394	IF122	Kalkulus	4	TIPE_KULIAH_TEORI	SHIFT_REGULER	[11010]	A	1307
1	3	0	2020130026	2	604	KU107	Character Building II	2	TIPE_KULIAH_TEORI	SHIFT_REGULER	[12209]	A	1310

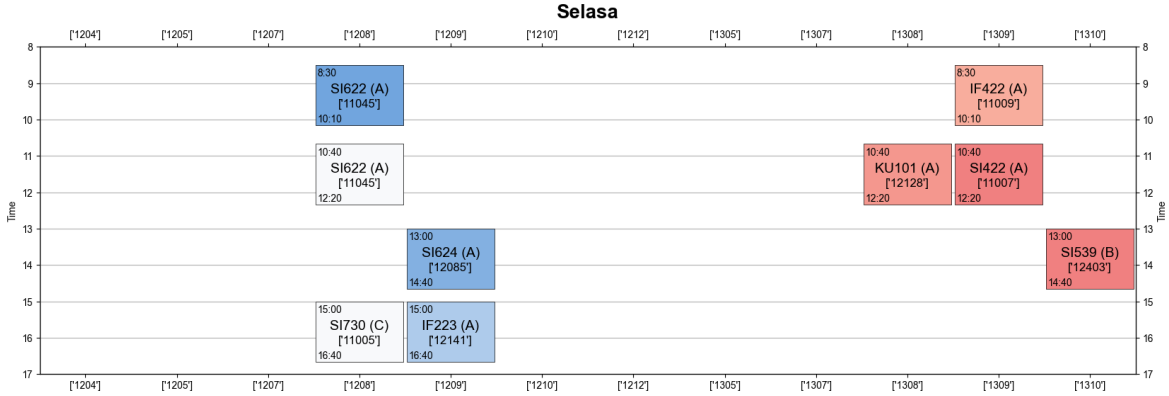
Gambar 2. Sisa pelanggaran batasan dari solusi terbaik penjadwalan kelas Reguler



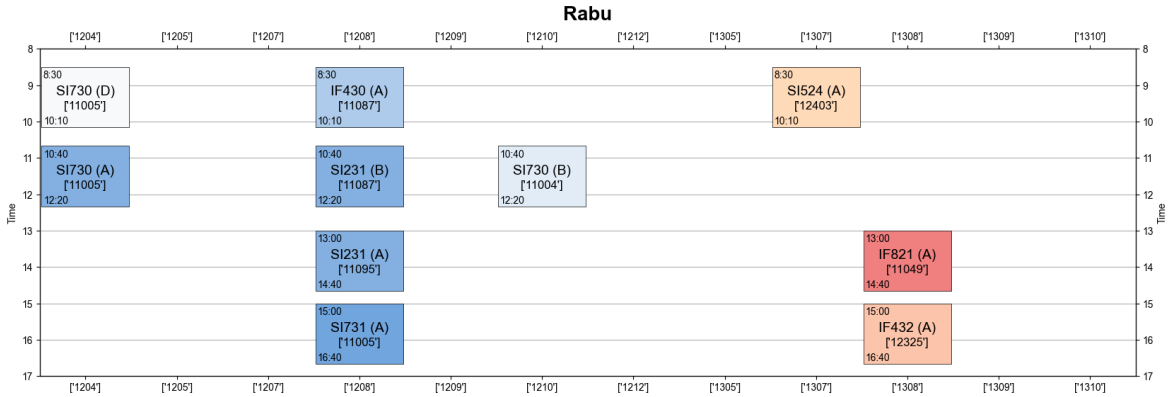
Gambar 3. Grafik proses penjadwalan kelas Reguler yang menghasilkan solusi terbaik



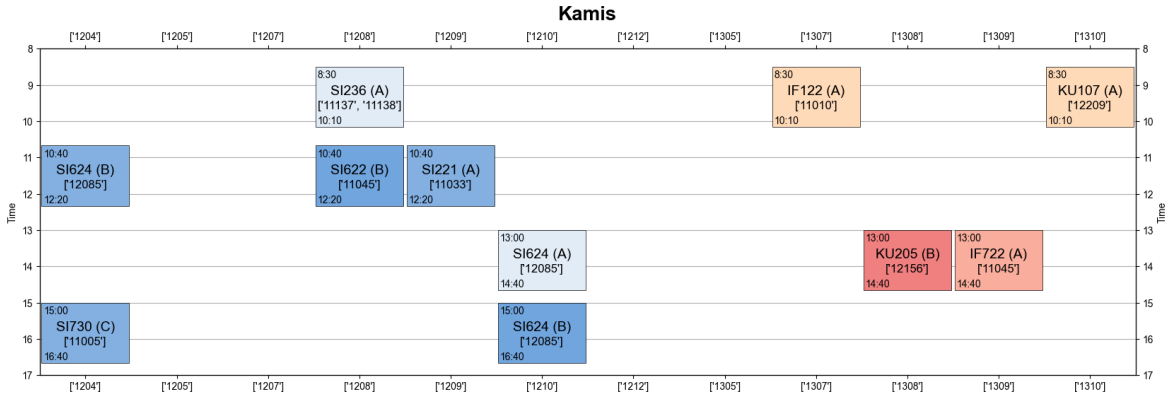
Gambar 4. Solusi terbaik jadwal kuliah hari Senin kelas Reguler



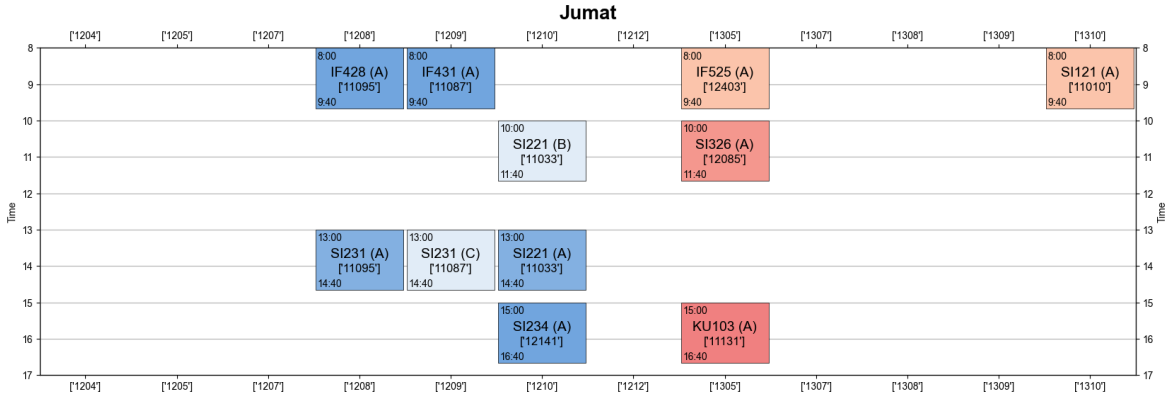
Gambar 5. Solusi terbaik jadwal kuliah hari Selasa kelas Reguler



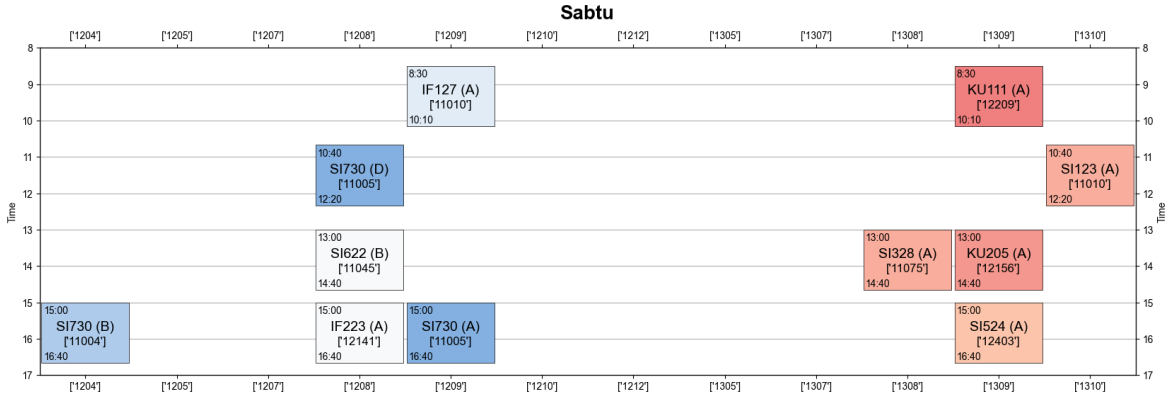
Gambar 6. Solusi terbaik jadwal kuliah hari Rabu kelas Reguler



Gambar 7. Solusi terbaik jadwal kuliah hari Kamis kelas Reguler



Gambar 8. Solusi terbaik jadwal kuliah hari Jumat kelas Reguler



Gambar 9. Solusi terbaik jadwal kuliah hari Sabtu kelas Reguler

b. Penjadwalan untuk kelas Karyawan saja.

Tabel 6. Hasil *plotting* kelas Karyawan

Data	**Jumlah**
Mahasiswa	114 orang
Kelas Teori	25 kelas
Kelas Praktik	11 kelas

Tabel 7. Hasil percobaan penjadwalan kelas Karyawan

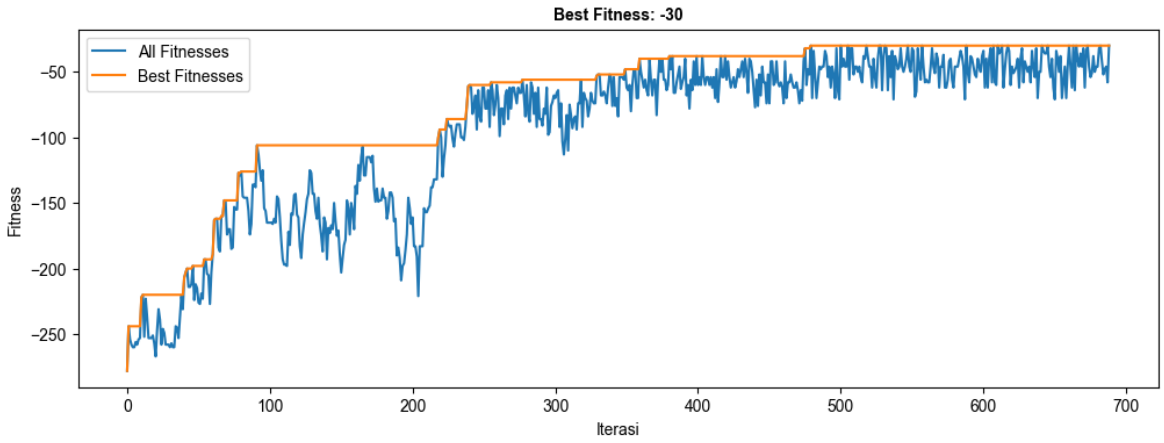
	Percobaan										$\bar{x}$
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	

Fitness Awal	-271	-224	-318	-400	-329	-356	-278	-334	-299	-419	-322,8
H1	0	0	0	0	0	0	0	0	0	0	0,0
H2	0	0	0	0	0	0	0	0	0	0	0,0
H3	0	0	0	0	0	0	0	0	0	0	0,0
H4	0	0	0	0	0	0	0	0	0	0	0,0
H5	0	0	0	0	0	0	0	0	0	0	0,0
H6	0	0	0	0	0	0	0	0	0	0	0,0
H7	0	0	0	0	0	0	0	0	0	0	0,0
H8	0	0	0	0	0	0	0	0	-2	0	-0,2
S1	-55	-34	-55	-42	-55	-34	-30	-62	-38	-58	-46,3
S2	0	0	0	0	0	0	0	0	0	0	0,0
Fitness Akhir	-55	-34	-55	-42	-55	-34	**-30**	-62	-40	-58	-46,5
Ulang	1	0	0	0	2	1	0	0	1	0	0,5
Iterasi	1327	688	688	688	1777	1215	688	688	1316	688	976,3
Durasi (s)	72,76	37,14	36,34	39,32	95,56	65,26	39,63	36,86	75,61	35,57	53,41

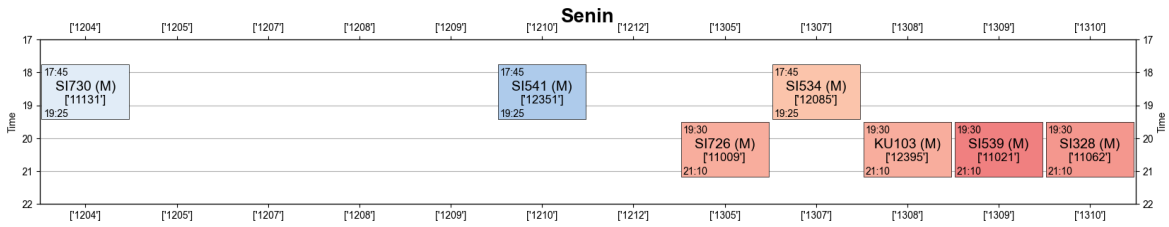
```
S1      : 30
Mahasiswa tidak boleh mendapat jadwal kelas lebih dari sekali dalam slot hari dan waktu yang sama.
```

	hari_id	timeslot	mahasiswa	counts	mata_kuliah_id	kode	nama_kuliah	sks	tipe_kuliah	shift	niks	kelas	ruang
0	0	5	2019110118	2	475	SI726	Etika Profesi	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11009]	M	1305
1	0	5	2019110118	2	480	SI328	Manajemen Stratejik	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11062]	M	1310
2	0	5	2021110059	2	98	KU103	Pancasila	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[12395]	M	1308
3	0	5	2021110059	2	480	SI328	Manajemen Stratejik	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11062]	M	1310
4	1	4	2022110085	2	454	SI422	Akuntansi Keuangan	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11021]	M	1305
5	1	4	2022110085	2	663	SI541	ENTERPRISE RESOURCES PLANNING LANJUT	4	TIPE_KULIAH_PRAKTIK	SHIFT_MALAM	[12351]	M	1210
6	1	4	2022110086	2	448	SI324	Manajemen Pemasaran	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11075]	M	1310
7	1	4	2022110086	2	450	SI326	Komunikasi Bisnis	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11062]	M	1308
8	1	5	2020110087	2	458	SI426	Akuntansi Menengah	4	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11107]	M	1309

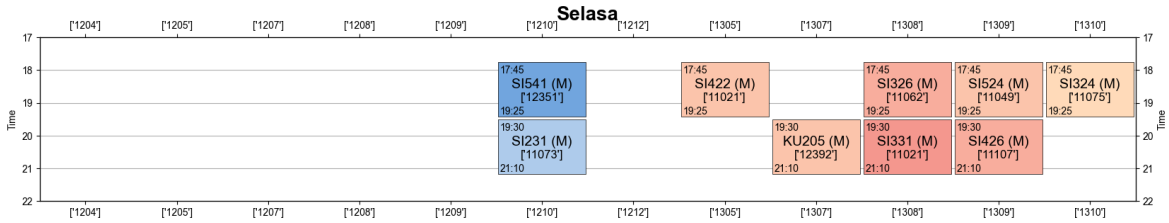
Gambar 10. Potongan sisa pelanggaran batasan dari solusi terbaik penjadwalan kelas Karyawan



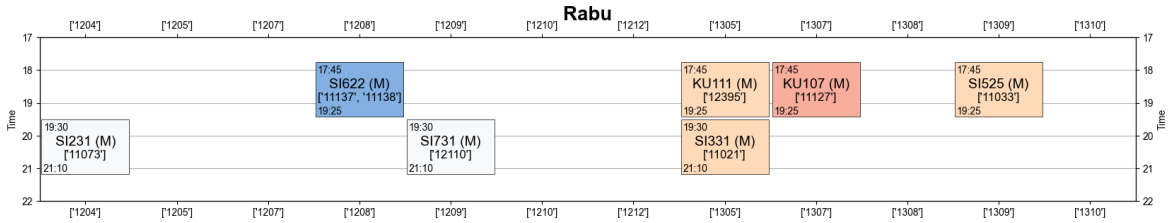
Gambar 11. Grafik proses penjadwalan kelas Karyawan yang menghasilkan solusi terbaik



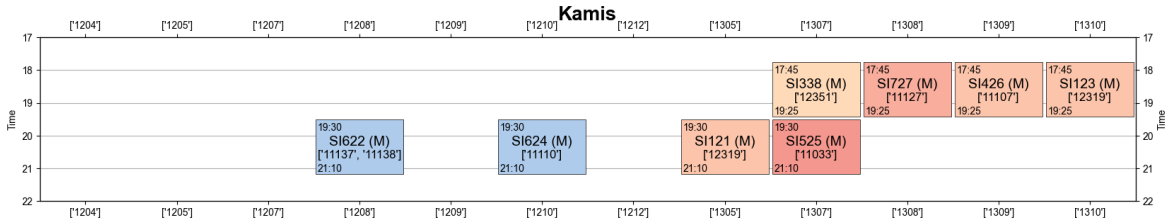
Gambar 12. Solusi terbaik jadwal kuliah hari Senin kelas Karyawan



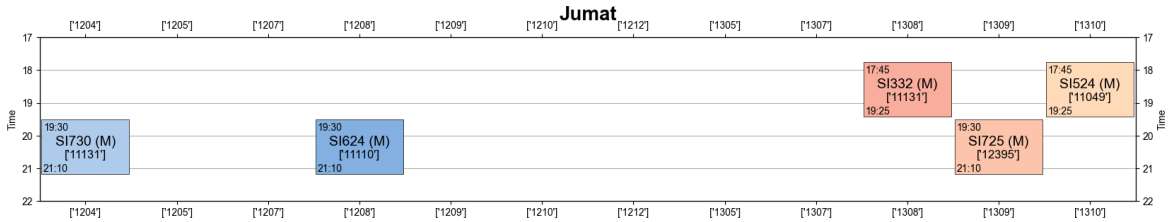
Gambar 13. Solusi terbaik jadwal kuliah hari Selasa kelas Karyawan



Gambar 14. Solusi terbaik jadwal kuliah hari Rabu kelas Karyawan



Gambar 15. Solusi terbaik jadwal kuliah hari Kamis kelas Karyawan



Gambar 16. Solusi terbaik jadwal kuliah hari Jumat kelas Karyawan

c. Penjadwalan untuk kelas Reguler dan Karyawan.

Tabel 8. Hasil *plotting* gabungan

Data	**Jumlah**
Mahasiswa	280 orang
Kelas Teori	51 kelas
Kelas Praktik	47 kelas

Tabel 9. Hasil percobaan penjadwalan gabungan

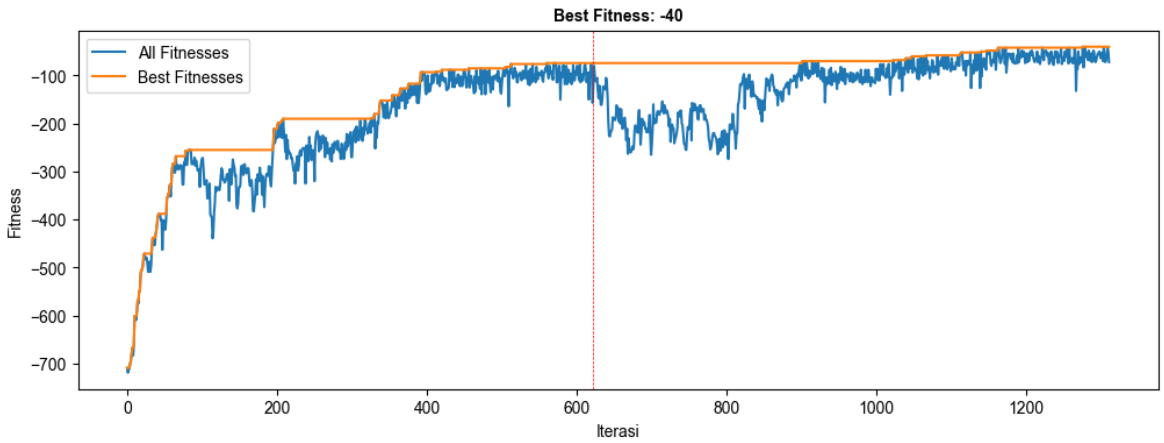
	Percobaan										$\bar{x}$
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Fitness Awal	-776	-841	-705	-737	-709	-585	-791	-857	-717	-674	-739,2
H1	0	0	0	0	0	0	0	0	0	0	0,0
H2	0	0	0	0	0	0	0	0	0	0	0,0
H3	0	0	0	0	0	0	0	0	0	0	0,0
H4	0	0	0	0	0	0	0	0	0	0	0,0
H5	0	0	0	0	0	0	0	0	0	0	0,0
H6	0	0	0	0	0	0	0	0	0	0	0,0
H7	0	0	0	0	0	0	0	0	0	0	0,0

H8	0	-2	0	-2	0	0	0	-2	0	0	-0,6
S1	-56	-79	-86	-63	-40	-70	-73	-74	-67	-62	-67
S2	0	0	0	0	0	0	0	0	0	0	0,0
Fitness Akhir	-56	-81	-86	-65	**-40**	-70	-73	-76	-67	-62	-67,6
Ulang	0	0	0	0	1	0	0	0	0	0	0,1
Iterasi	688	688	688	688	1310	688	688	688	688	688	750,2
Durasi (s)	42,61	42,89	42,42	42,79	80,91	42,38	42,77	48,13	51,65	46,87	48,34

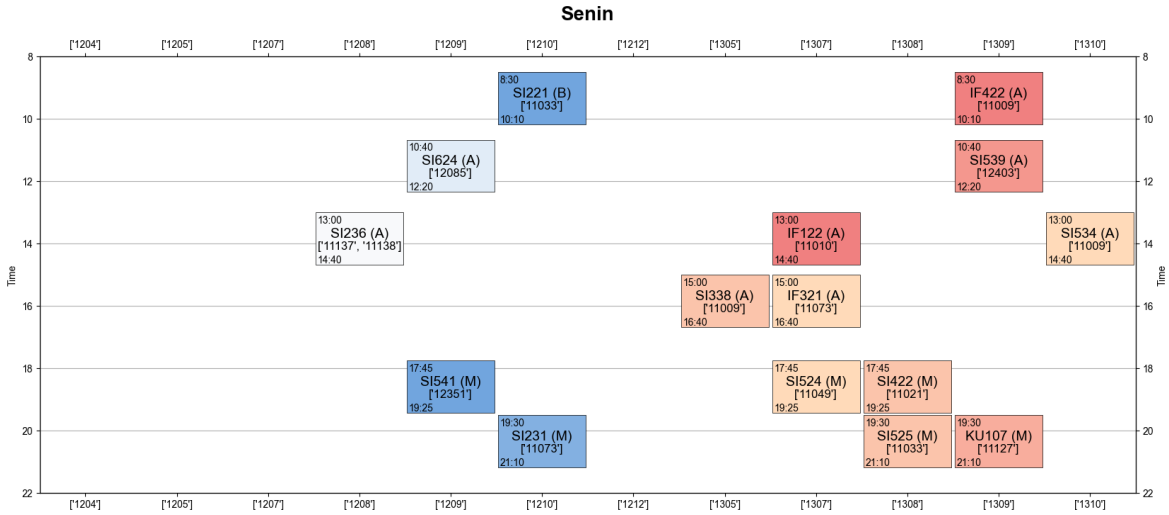
```
S1      : 40
Mahasiswa tidak boleh mendapat jadwal kelas lebih dari sekali dalam slot hari dan waktu yang sama.
```

	hari_id	timeslot	mahasiswa	counts	mata_kuliah_id	kode	nama_kuliah	sks	tipe_kuliah	shift	niks	kelas	ruang
0	0	4	2022110085	2	454	SI422	Akuntansi Keuangan	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11021]	M	1308
1	0	4	2022110085	2	663	SI541	ENTERPRISE RESOURCES PLANNING LANJUT	4	TIPE_KULIAH_PRAKTIK	SHIFT_MALAM	[12351]	M	1209
2	0	5	2021110064	2	443	SI231	Pengolahan Basis Data	4	TIPE_KULIAH_PRAKTIK	SHIFT_MALAM	[11073]	M	1210
3	0	5	2021110064	2	463	SI525	Rekayasa Perangkat Lunak	4	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11033]	M	1308
4	1	4	2019110023	2	469	SI622	Pemrograman Berbasis Web	4	TIPE_KULIAH_PRAKTIK	SHIFT_MALAM	[11137, 11138]	M	1208
5	1	4	2019110023	2	476	SI727	Kecakapan Antar Personal	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[11127]	M	1310
6	1	4	2022110055	2	469	SI622	Pemrograman Berbasis Web	4	TIPE_KULIAH_PRAKTIK	SHIFT_MALAM	[11137, 11138]	M	1208
7	1	4	2022110055	2	651	SI338	PROSES BISNIS	2	TIPE_KULIAH_TEORI	SHIFT_MALAM	[12351]	M	1307
8	1	5	2019110023	2	469	SI622	Pemrograman Berbasis Web	4	TIPE_KULIAH_PRAKTIK	SHIFT_MALAM	[11137, 11138]	M	1208

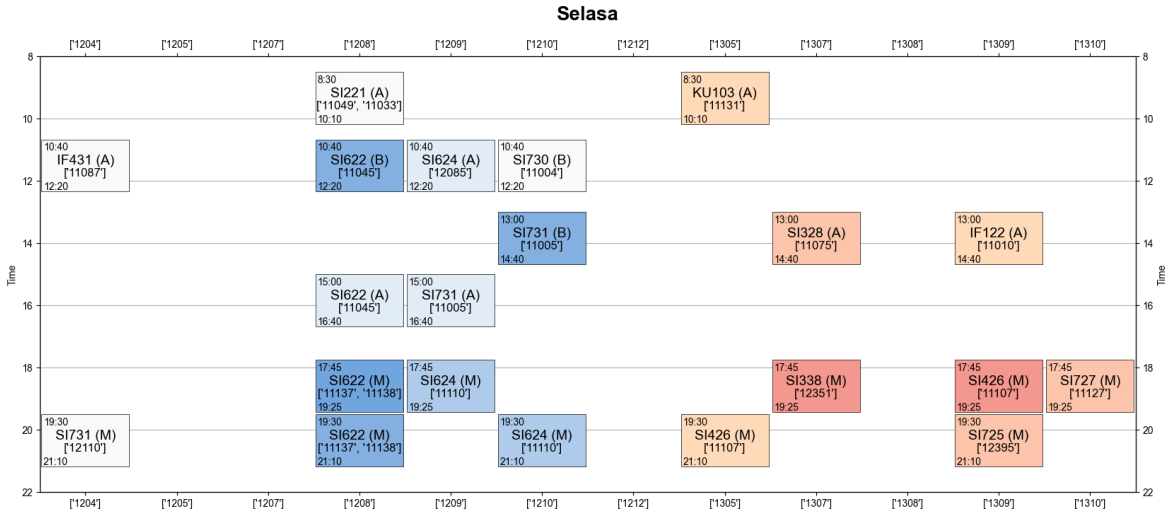
Gambar 17. Potongan sisa pelanggaran batasan dari solusi terbaik penjadwalan gabungan



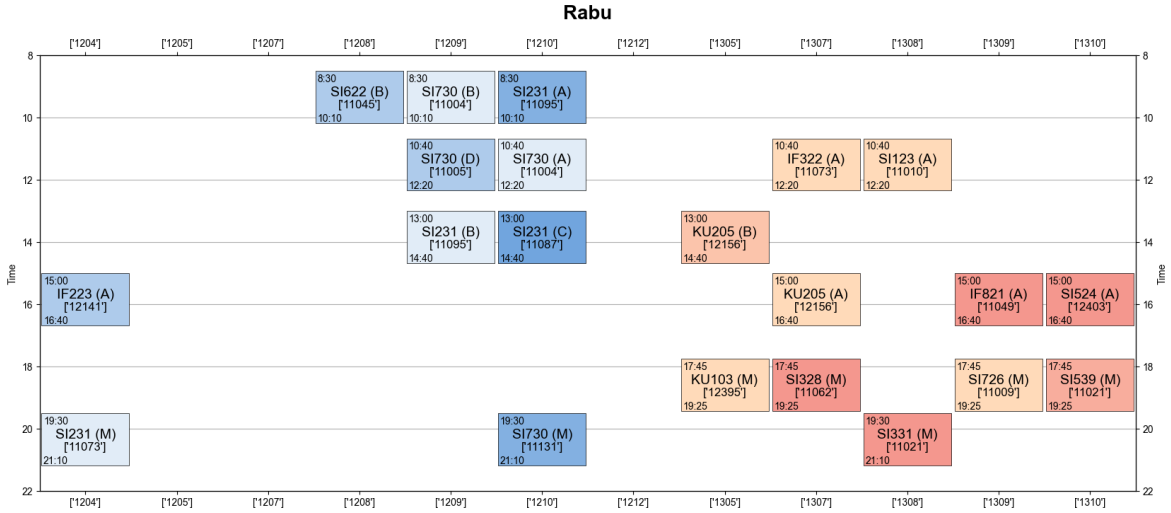
Gambar 18. Grafik proses penjadwalan gabungan yang menghasilkan solusi terbaik



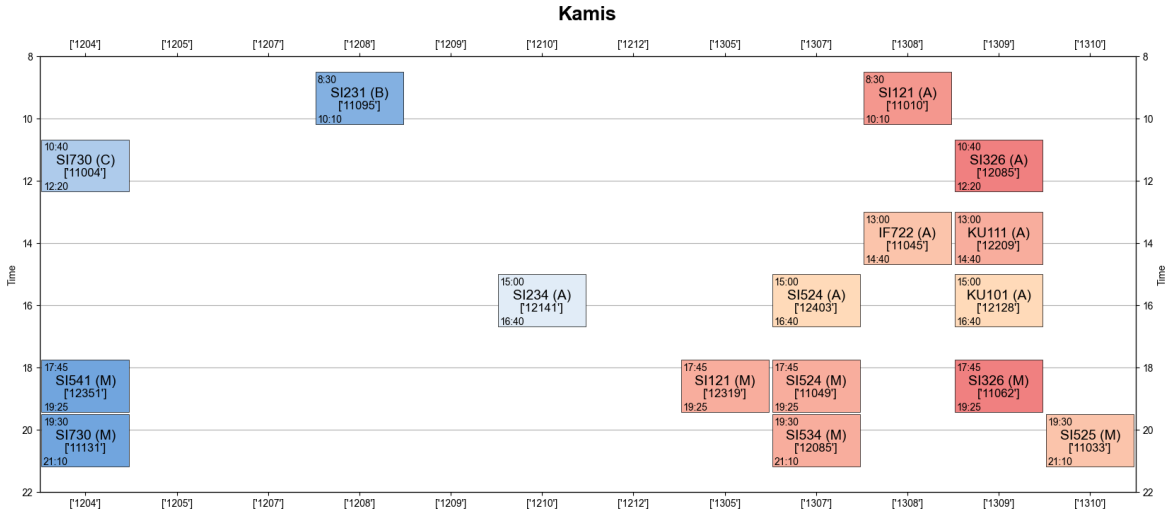
Gambar 19. Solusi terbaik jadwal kuliah hari Senin gabungan



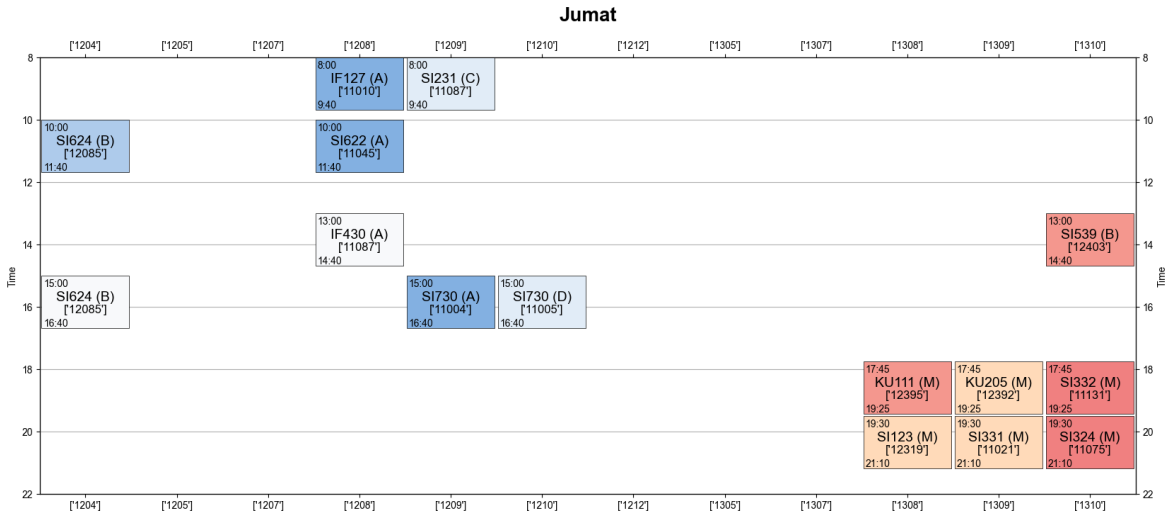
Gambar 20. Solusi terbaik jadwal kuliah hari Selasa gabungan



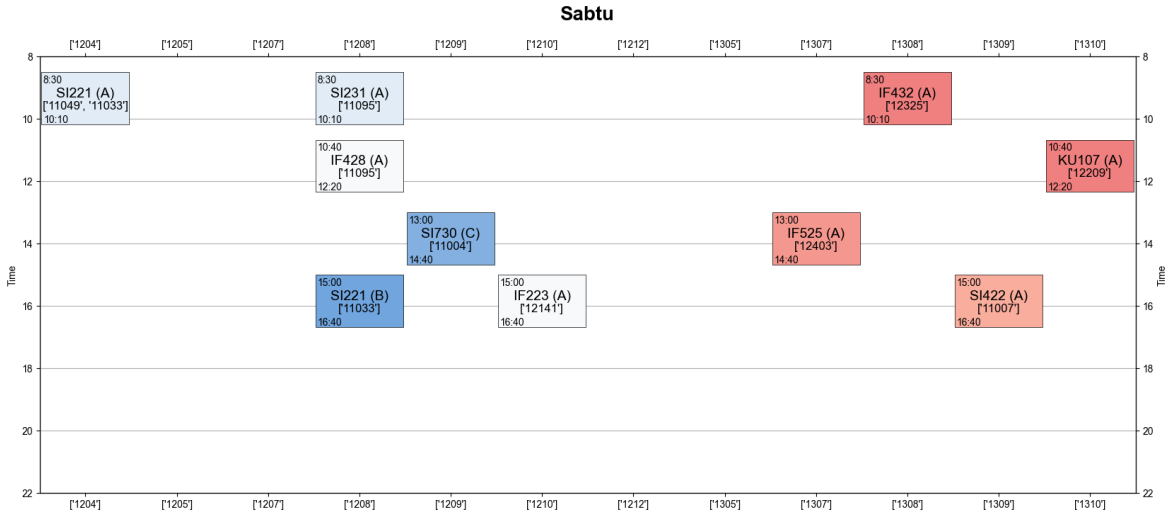
Gambar 21. Solusi terbaik jadwal kuliah hari Rabu gabungan



Gambar 22. Solusi terbaik jadwal kuliah hari Kamis gabungan



Gambar 23. Solusi terbaik jadwal kuliah hari Jumat gabungan



Gambar 24. Solusi terbaik jadwal kuliah hari Sabtu gabungan

4.4. Pembahasan Hasil Penjadwalan Kuliah

Penjadwalan dengan algoritma SA yang telah disesuaikan menghasilkan solusi-solusi terbaik untuk masing-masing ruang lingkup penjadwalan, yaitu kelas Reguler, Karyawan, dan gabungan keduanya. Tampak bahwa penjadwalan untuk kelas Reguler dapat memberikan solusi terbaik yang hampir ideal berdasarkan pemenuhan batasan-batasan yang sudah ditentukan sebelumnya, ditunjukkan oleh Gambar 3, dibandingkan dengan kelas Karyawan yang ditunjukkan oleh Gambar 11. Hal ini dipengaruhi oleh ketersediaan slot waktu yang berbeda, di mana kelas Reguler memiliki slot waktu yang lebih banyak daripada kelas Karyawan.

Sisa pelanggaran batasan dari solusi terbaik penjadwalan gabungan, ditunjukkan oleh Gambar 17, tampak mendekati gabungan sisa pelanggaran solusi terbaik penjadwalan kelas Reguler (Gambar 2) dan Karyawan (Gambar 10), walaupun tentu angkanya tidak dapat dijumlahkan secara langsung. Selisih angka pelanggaran yang dihasilkan antara penjadwalan gabungan dan masing-masing kelas dapat dipengaruhi oleh adanya data dosen yang mengajar di kedua jenis kelas tersebut dan/atau data mahasiswa yang mengambil dua jenis kelas pada semester yang diambil, sehingga meningkatkan kompleksitas untuk menemukan solusi jadwal yang terbaik.

Secara keseluruhan, algoritma SA yang disesuaikan dapat menghasilkan solusi penjadwalan terbaik relatif cepat, sambil mengupayakan pemenuhan batasan-batasan semaksimal mungkin. Adapun rata-rata usaha penemuan solusi jadwal terbaik untuk kelas Reguler membutuhkan waktu 65,86 detik dengan capaian *fitness* -16,2 (Tabel 5), kelas Karyawan membutuhkan waktu 53,41 detik dengan capaian *fitness* -46,5 (Tabel 7), dan gabungan membutuhkan waktu 48,34 detik dengan capaian *fitness* -67,6 (Tabel 9). Hal tersebut mengindikasikan bahwa dengan menggabungkan ruang lingkup penjadwalan, proses penemuan solusi memakan waktu lebih cepat daripada menjadwalkannya secara individual, tetapi harus berkompromi dengan keidealan (*fitness*) solusi yang diperoleh.

5. KESIMPULAN

Berdasarkan pemaparan hasil penelitian yang diungkapkan, maka disimpulkan bahwa algoritma *Simulated Annealing* (SA) dapat disesuaikan dengan kebutuhan studi kasus dan mampu menghasilkan solusi jadwal terbaik dalam berbagai ruang lingkup penjadwalan. Kodifikasi sebagian batasan keras sebagai bagian penyesuaian algoritma penjadwalan turut andil dalam mempermudah evaluasi solusi. Fleksibilitas konfigurasi penjadwalan juga dapat memberikan ragam solusi dan kompromi antara kecepatan proses dengan keidealan solusi.

Penyesuaian algoritma SA tersebut dapat memenuhi semua batasan-batasan keras dan lunak semaksimal mungkin sehingga menghasilkan solusi jadwal terbaik dalam waktu yang relatif singkat meskipun menggunakan kombinasi data yang kompleks. Solusi jadwal terbaik masih menyisakan pelanggaran batasan-batasan lunak karena dipengaruhi oleh keterkaitan data dosen, mahasiswa, dan jenis kelas yang ditetapkan. Walaupun demikian, data-data pelanggaran tersebut dapat ditindaklanjuti dengan lebih mudah karena sudah terlampir bersama deskripsi pelanggaran yang dimaksud dan jumlah pelanggaran yang relatif sedikit untuk ditangani.

DAFTAR PUSTAKA

[1] M. C. Chen, S. N. Sze, S. L. Goh, N. R. Sabar dan G. Kendall, “A Survey of University Course Timetabling: Perspectives, Trends and Opportunities,” *IEEE Access,* vol. 9, pp. 106515-106529, 2021.

[2] D. Kusumawardani, A. Muklason dan V. A. Supoyo, “Examination Timetabling Automation and Optimization using Greedy-Simulated Annealing Hyper-heuristics Algorithm,” dalam *12th International Conference on Information & Communication Technology and System (ICTS)*, Surabaya, 2019.

[3] E. Norgren dan J. Jonasson, “Investigating a Genetic Algorithm-Simulated Annealing Hybrid Applied to University Course Timetabling Problem: A Comparative Study Between Simulated Annealing Initialized with Genetic Algorithm, Genetic Algorithm and Simulated Annealing,” Digitala Vetenskapliga Arkivet, 2016.

[4] S. S. Rao, Engineering Optimization Theory and Practice, Hoboken, New Jersey: John Wiley & Sons Ltd, 2020.

[5] H. A. E. Oliveira Junior, L. Ingber, A. Petraglia, M. R. Petraglia dan M. A. S. Machado, “Stochastic Global Optimization and Its Applications with Fuzzy Adaptive Simulated Annealing,” dalam *Intelligent systems reference library*, vol. 35, Springer-Verlag Berlin Heidelberg, 2012, pp. 3-10.

[6] M. J. Kochenderfer dan T. A. Wheeler, Algorithms for Optimization, Cambridge, Massachusetts: The MIT Press, 2019.

[7] H. Babaei, J. Karimpour dan A. Hadidi, “A survey of approaches for university course timetabling problem,” *Computers & Industrial Engineering,* vol. 86, pp. 43-59, 2015.

[8] D. Delahaye, S. Chaimatanan dan M. Mongeau, "Simulated Annealing: From Basics to Applications," dalam *International Series in Operations Research & Management Science*, Springer, 2019, pp. 1-35.