BAB III
METODE PENELITIAN

Bab ini membahas metodologi dan perancangan yang digunakan dalam penyelesaian masalah penjadwalan *flexible job shop with due windows* dengan menggunakan GASA. Pembahasan ini mencakup deskripsi masalah, tahapan penelitian, formulasi model optimisasi dan teknik penyelesaian.

3.1 Deskripsi Masalah

Penelitian ini membahas mengenai masalah penjadwalan *flexible job shop with due windows*. Pada masalah ini, terdapat fleksibilitas dalam alokasi mesin untuk setiap operasi dalam suatu *job* serta terdapat *due windows* atau interval waktu yang diizinkan untuk menyelesaikan suatu *job*. Setiap *job* harus selesai dalam interval waktu (*due windows*) yang telah ditentukan, di mana interval waktu tersebut berupa waktu paling awal suatu *job* dapat selesai dan batas waktu akhir suatu *job* harus selesai.

Dalam sebuah perusahaan, penyelesaian suatu *job* yang terlalu cepat (*earliness*) dapat menyebabkan kerugian, seperti meningkatnya risiko kerusakan barang dan keterbatasan ruang gudang. Sebaliknya, penyelesaian *job* yang terlambat (*tardiness*) dapat menyebabkan ketidakpuasan pelanggan dan gangguan pada rantai pasok. Oleh karena itu, tujuan utama dari penelitian ini adalah untuk meminimumkan penalti yang timbul akibat penyelesaian *job* di luar interval waktu (*due windows*) yang telah ditentukan, baik untuk *job* yang selesai lebih awal dari batas bawah *due windows* (*earliness*) maupun *job* yang melewati batas atas *due windows* (*tardiness*) sehingga setiap *job* dapat selesai tepat dalam interval waktu yang telah ditentukan.

Misalkan terdapat n *job*, yaitu $J = \{J_1, J_2, \dots, J_n\}$ dan m mesin, yaitu $M = \{M_1, M_2, \dots, M_m\}$. Setiap *job* terdiri dari beberapa operasi yang urutannya telah ditentukan sebelumnya, dan setiap operasi memerlukan satu mesin untuk mengerjakannya. Masalah utama dalam penelitian ini adalah bagaimana menentukan urutan *job* dan alokasi mesin sehingga penalti *earliness* dan *tardiness*

dapat diminimalkan agar setiap *job* dapat selesai tepat waktu sesuai dengan interval waktu yang telah ditentukan. GASA akan diimplementasikan untuk menyelesaikan masalah penjadwalan *flexible job shop with due windows* ini.

Pada penelitian ini penyelesaian masalah *flexible job shop with due windows* menggunakan GASA akan diimplementasikan untuk menyelesaikan masalah penjadwalan produksi sepatu dan sandal. Produksi sepatu dan sandal memerlukan beberapa tahapan utama, yaitu penggambaran pola pada bahan, pemotongan bahan, penjahitan, *lasting*, dan *finishing*. Tahapan ini dipandang sebagai operasi. Selain itu, terdapat beberapa jenis sepatu dan sandal yang harus dibuat. Jenis sepatu ini dapat dipandang sebagai *job*. Pada setiap *job*, perusahaan menambahkan batasan waktu untuk menyelesaikan *job* tersebut. Setiap tahapan pembuatan sepatu dan sandal, tersedia beberapa pilihan mesin yang dapat dipilih secara bebas sesuai dengan mesin yang tersedia pada operasi atau tahapan tersebut. Dengan demikian masalah pembuatan produksi sepatu ini dapat dipandang sebagai masalah *flexible job shop with due windows.*

3.2 Tahapan Penelitian

Penelitian ini terdiri dari beberapa tahapan yaitu sebagai berikut.

1. Studi Pustaka

   Studi pustaka yang dilakukan pada penelitian ini adalah dengan mempelajari teori-teori yang berkaitan dengan *flexible job shop with due windows* dan GASA yang diperoleh melalui jurnal nasional, jurnal internasional, buku, dan skripsi.

2. Pengumpulan Data

   Data yang diambil pada penelitian ini terdiri dari data *job* (pesanan barang), data operasi atau tahapan setiap *job* (pesanan barang), data alternatif mesin, data waktu proses setiap alternatif mesin, dan data interval waktu (*due windows*) setiap *job* (pesanan barang). Data-data tersebut diperoleh melalui wawancara dengan pekerja dan perhitungan waktu secara manual.

3. Pembangunan Model Optimisasi

   Tahap ini akan membangun model optimisasi dari masalah penjadwalan

flexible job shop with due windows dengan terlebih dahulu mendefinisikan himpunan, indeks, dan parameter yang digunakan pada model optimisasi.

4. Penyelesaian Model

   Pada tahap ini, masalah penjadwalan *flexible job shop with due windows* dan akan diselesaikan dengan GASA.

5. Validasi

   Sebelum dilakukan implementasi, dilakukan terlebih dahulu tahapan validasi untuk menguji model optimisasi dan GASA yang digunakan sudah valid atau tidak. Validasi dilakukan dengan membandingkan solusi optimal yang diperoleh dengan menggunakan bahasa pemrograman Python dengan solusi optimal dari perhitungan secara manual dari suatu kasus *flexible job shop with due windows* berukuran kecil. Apabila hasilnya valid atau solusi yang diperoleh sama, maka tahapan akan dilanjutkan pada tahap implementasi. Apabila hasilnya tidak valid atau solusi yang diperoleh tidak sama, maka tahapan akan diulang kembali dari tahap pemodelan.

6. Implementasi

   Model dan teknik penyelesaian yang telah valid, akan diimplementasikan untuk menyelesaikan masalah penjadwalan *flexible job shop with due windows* pada suatu perusahaan produksi sepatu dan sandal di Kota Bandung.

7. Penarikan Kesimpulan

   Pada tahap ini akan ditarik kesimpulan berdasarkan hasil implementasi dan analisis kinerja GASA dalam menyelesaikan masalah penjadwalan *flexible job shop with due windows*.

3.3 Model Optimisasi

Model optimisasi *flexible job shop with due windows* diformulasikan dengan merujuk pada model optimisasi Song (2012) menggunakan asumsi-asumsi sebagai berikut.

1. Seluruh *job* datang pada saat waktu 0 (t = 0).
2. *Job* bersifat independen satu sama lain, artinya setiap *job* tidak saling bergantung satu sama lain sehingga urutan pengerjaan dapat disesuaikan tanpa

adanya batasan ketergantungan antar *job*.

3. Waktu pengaturan terhadap mesin diabaikan.
4. Waktu perpindahan antar operasi diabaikan.
5. Setiap mesin hanya dapat melakukan satu operasi dalam suatu interval waktu.
6. Hanya satu operasi dari *job* yang sama dapat diproses dalam suatu interval waktu.
7. Untuk setiap *job*, urutan operasi telah ditentukan sebelumnya.
8. Bahan dan desain pola sepatu dan sandal sudah tersedia pada waktu 0 (t = 0).

Adapun tahapan formulasi model optimisasi ini adalah sebagai berikut.

1. Pendefinisian himpunan dan parameter.

   Himpunan-himpunan model optimisasi terdiri dari himpunan *job*, himpunan operasi, himpunan mesin, himpunan operasi dari setiap *job*, dan himpunan mesin yang dapat memproses operasi. Selain itu terdapat parameter yang digunakan pada model optimisasi yang terdiri dari operasi ke-*i* dari *job j*, waktu penyelesaian operasi, waktu proses operasi, waktu mulai operasi, batas atas waktu penyelesaian *job,* batas bawah waktu penyelesaian *job*, bobot penalti untuk *earliness,* bobot penalti untuk *tardiness, earliness* untuk *job j,* dan *tardiness* untuk *job j*.

2. Pendefinisian variabel keputusan

   Variabel keputusan model didefinisikan untuk mengidentifikasi operasi dikerjakan pada suatu mesin tertentu.

3. Pendefinisian fungsi tujuan

   Fungsi tujuan dari model optimisasi didefinisikan untuk meminimumkan penalti terhadap waktu penyelesaian *job*, baik untuk *job* yang selesai lebih awal (*earliness*) maupun *job* yang terlambat (*tardiness*) sehingga setiap *job* dapat selesai sesuai dengan interval waktu (*due windows*) yang telah ditentukan.

4. Menetapkan fungsi kendala

   Kendala-kendala model optimisasi menjamin bahwa:

   1. Setiap operasi dikerjakan oleh satu mesin;
   2. Waktu penyelesaian keseluruhan dari *job j* diambil dari waktu penyelesaian maksimum dari seluruh operasinya;

3. Urutan operasi dari suatu *job* tidak dilanggar;
4. *Earliness* harus bernilai positif;
5. *Tardiness* harus bernilai positif.

5. Menentukan batasan model

   Batasan dari model optimisasi menjamin bahwa seluruh variabel bernilai non negatif.

3.4 Teknik Penyelesaian

Penelitian ini menggunakan GASA untuk menyelesaikan masalah *flexible job shop with due windows*. GASA merupakan penggabungan dari algoritma genetika dan *simulated annealing*. Kedua algoritma tersebut memiliki kekurangan tersendiri. Algoritma genetika memiliki kekurangan, yaitu ketika solusi yang diperoleh terindikasi bersifat konvergen prematur. Sedangkan *simulated annealing* memiliki kekurangan yaitu, *simulated annealing* hanya dapat menyimpan satu solusi terbaik pada tiap iterasinya dan mengabaikan solusi lainnya yang memungkinkan menghasilkan nilai yang lebih baik.

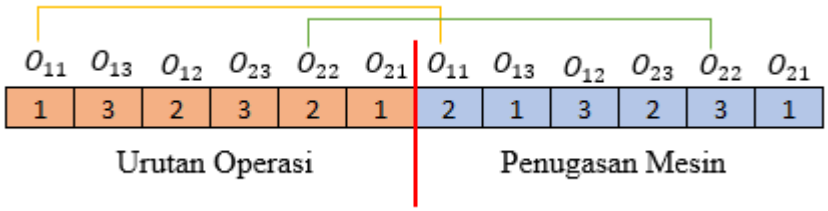
Proses pencarian solusi optimum menggunakan GASA merujuk pada penelitian Firmansyah (2020) dan Song (2012). Secara garis besar, pencarian solusi ini diawali dengan melakukan proses pencarian solusi menggunakan algoritma genetika pada solusi awal. Pada tahap ini, proses dimulai dengan inisialisasi populasi dan evaluasi *fitness*, dilanjutkan dengan proses evolusi yang terdiri dari seleksi individu, *crossover*, dan mutasi. Proses tersebut diulang hingga mencapai kriteria pemberhentian algoritma genetika, sehingga diperoleh solusi terbaik dari algoritma genetika. Setelah diperoleh solusi terbaik dari algoritma genetika, dilanjutkan dengan proses pencarian solusi menggunakan *simulated annealing* dengan memodifikasi solusi tersebut untuk eksplorasi lebih lanjut. Tahap akhir dari proses ini adalah penurunan suhu. Jika suhu masih lebih tinggi dari suhu akhir, maka kembali pada pencarian solusi dengan menggunakan algoritma genetika. Proses ini akan diulang hingga suhu akhir tercapai. Langkah penyelesaian masalah *flexible job shop with due windows* menggunakan GASA adalah sebagai berikut.

1. Inisialisasi Parameter Awal

   Parameter yang digunakan pada GASA terdiri dari ukuran populasi ($Pop\ size$), generasi maksimum (*max gen*), *mutation rate* (M_r), *crossover rate* (C_r), suhu awal (T_0), suhu akhir (T_f), dan laju penurunan suhu (α).

2. Representasi Kromosom

   Setiap individu pada algoritma genetika dapat merepresentasikan solusi potensial. Pada FJSP kromosom dibagi menjadi dua bagian utama, yaitu bagian urutan operasi (*operation sequence*) dan bagian penugasan mesin (*machine assignment*) (Karimi, dkk., 2012 dalam Chen, dkk., 2020). Solusi yang diberikan berupa pengaturan urutan operasi dan penugasan mesin yang terlibat pada proses produksi. Contoh representasi kromosom dapat dilihat pada Gambar 3.2.



Gambar 3.1 Contoh Representasi Kromosom

Gambar 3.1 menunjukkan bahwa terdapat 6 operasi dengan panjang kromosom adalah 12 atau dua kali lebih panjang dari jumlah operasinya. Hal ini disebabkan karena setiap operasi diwakili oleh dua elemen dalam kromosom, yaitu urutan operasi dan penugasan mesin yang akan mengerjakan operasi tersebut.

Urutan operasi dapat dibaca dari kiri ke kanan. Setiap angka pada kromosom merepresentasikan operasi yang harus dilakukan oleh sebuah *job*. Misalkan pada Gambar 3.2 angka 1 pertama pada urutan operasi menunjukkan bahwa operasi pertama dari *job* 1 (J_1) akan dikerjakan oleh mesin M_2 yang direpresentasikan oleh angka 2 pada penugasan mesin. Angka 1 kedua menunjukkan bahwa operasi kedua dari *job* 1 (J_1) yang akan dikerjakan oleh mesin M_1 dan direpresentasikan oleh angka 1 pada penugasan mesin. Berdasarkan penjelasan tersebut, urutan operasi pada mesin M_1 adalah $O_{13} \rightarrow$

O_{21}, pada mesin M_2 adalah $O_{11} \rightarrow O_{23}$, dan pada mesin M_3 adalah $O_{21} \rightarrow O_{22}$. Selain itu urutan operasi tersebut dapat diinterpretasikan sebagai berikut:

$$\{(O_{11}, M_2), (O_{12}, M_3), (O_{13}, M_1), (O_{21}, M_1), (O_{22}, M_3), (O_{23}, M_2)\}.$$

3. Inisialisasi Populasi

Pembangkitan populasi awal dilakukan secara acak dengan memperhatikan bahwa setiap operasi dikerjakan oleh mesin yang bersesuaian dan setiap kromosomnya mewakili urutan operasi yang berbeda serta penugasan mesin untuk menyelesaikan *job* dengan penalti *earliness* dan *tardiness* yang minimum. Banyaknya populasi awal ditentukan dengan parameter ukuran populasi ($Pop\ size$).

4. Perhitungan Nilai *Fitness*

Nilai *fitness* digunakan untuk menentukan kualitas dari suatu individu dalam populasi yang mencerminkan seberapa baik individu tersebut memenuhi kriteria atau tujuan optimasi yang telah ditetapkan. Nilai *fitness* setiap individu dalam populasi dihitung berdasarkan tujuan untuk meminimalkan total penalti *earliness* dan *tardiness* berdasarkan interval waktu penyelesaian *job* (*due windows*). Oleh sebab itu, fungsi *fitness* pada penelitian ini adalah sebagai berikut:

$$fitness = \frac{1}{1 + \left(w_E \sum_{j \in J} E_j + w_T \sum_{j \in J} T_j\right)}$$

Berdasarkan fungsi *fitness* tersebut, semakin tinggi nilai *fitness,* maka semakin baik kualitas solusi yang dihasilkan karena total penalti *earliness* dan *tardiness* menjadi semakin kecil. Sebaliknya, semakin rendah nilai *fitness*, maka semakin buruk kualitas solusi yang dihasilkan karena total penalti *earliness* dan *tardiness* menjadi semakin besar. Dengan demikian, terdapat hubungan terbalik antara nilai *fitness* dan total penalti *earliness* dan *tardiness*.

5. Seleksi

Seleksi dilakukan dengan metode seleksi *binary tournament* yang digunakan untuk memilih individu-individu dari populasi dan selanjutnya akan dilakukan operasi reproduksi (*crossover* dan mutasi). Langkah-langkah dari metode seleksi *binary tournament* adalah sebagai berikut.

a) Pilih dua individu secara acak dari populasi, misalkan individu 1 dan individu 2. Ulangi pengacakan sampai sebanyak jumlah populasi awal dan individu 1 ≠ individu 2.
b) Hitung nilai *fitness* dari kedua individu tersebut, lalu bandingkan nilai *fitness*-nya.
c) Pilih individu dengan nilai *fitness* tertinggi.

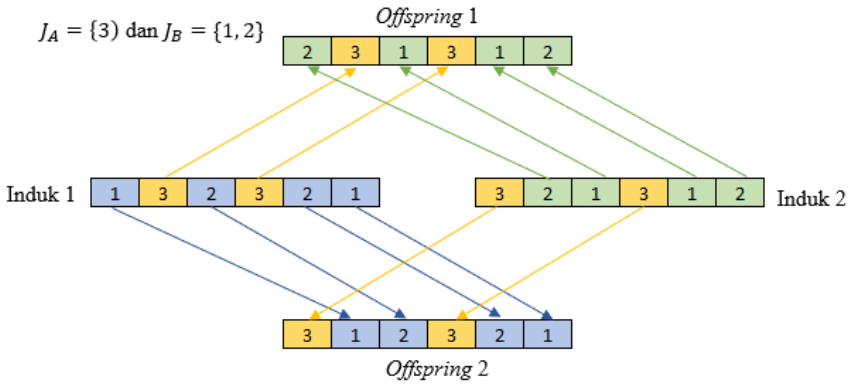
6. Crossover

Crossover dilakukan untuk mendapatkan individu baru dengan melibatkan dua induk sehingga menghasilkan kromosom baru (*offspring*). Dua induk tersebut diperoleh dengan memilih dua individu secara acak dari hasil seleksi. Setelah itu, bangkitkan bilangan acak antara 0 dan 1. Jika nilai bilangan acak kurang dari nilai *crossover rate* (C_r) maka kedua individu tersebut akan mengalami proses *crossover.* Jika nilai bilangan acak lebih besar atau sama dengan nilai *crossover rate* (C_r) maka kembali memilih dua individu secara acak dari hasil seleksi. Karena pada setiap kromosom terdapat dua bagian utama yaitu, urutan operasi dan penugasan mesin, maka digunakan dua metode *crossover* yaitu metode *precendence operation crossover* (POX) yang digunakan pada bagian urutan operasi dan metode *crossover* dua titik (*two point crossover*) yang digunakan pada bagian penugasan mesin.

Langkah-langkah metode POX yang akan digunakan pada bagian urutan operasi antara lain:

a) Pilih dua induk yang memiliki nilai *fitness* tertinggi, kemudian bagi set *job* (*J*) menjadi dua subset secara acak, misalkan J_A dan J_B dengan $J_A \cup J_B = J$ dan $J_A \cap J_B = \emptyset$.
b) Ambil elemen J_A lalu tempatkan pada *offspring* 1 pada posisi yang sama seperti pada induk 1. Isi posisi kosong di *offspring* 1 dengan mengikuti urutan elemen J_B seperti pada induk 2.
c) Ambil elemen J_A lalu tempatkan pada *offspring* 2 pada posisi yang sama seperti pada induk 2. Isi posisi kosong di *offspring* 1 dengan mengikuti urutan elemen J_B seperti pada induk 1.

Sebagai contoh proses seleksi pada bagian operasi dengan menggunakan *Precendence Operation Crossover* (POX) dapat dilihat pada Gambar 3.2.



Gambar 3.2 Contoh *Precendence Operation Crossover* (POX) pada Bagian Urutan Operasi

Adapun langkah-langkah metode *crossover* dua titik yang digunakan pada bagian penugasan mesin antara lain:

a) Pilih dua kromosom induk yang akan dilakukan *crossover*. Contoh kromosom induk 1 dan kromosom induk 2 dapat dilihat pada Gambar 3.3.

Induk 1 (MS)	2	1	3	2	3	1
Induk 2 (MS)	1	3	1	2	3	2

Gambar 3.3 Contoh Kromosom Induk pada Proses *Crossover* Dua Titik

b) Tetapkan dua titik secara acak pada masing-masing kromosom. Contoh pemilihan dua titik acak pada kromosom induk dapat dilihat pada Gambar 3.4.

Induk 1 (MS)	2	1	3	2	3	1
Induk 2 (MS)	1	3	1	2	3	2

Gambar 3.4 Contoh Titik Acak Kromosom Induk pada Proses *Crossover* Dua Titik

c) Tukar elemen acak di antara dua titik tersebut pada masing-masing induk sehingga membentuk keturunan (*offspring*) baru. Contoh *offspring* dapat dilihat pada Gambar 3.5.

Offspring 1 (MS)	2	1	1	2	3	1
Offspring 2 (MS)	1	3	3	2	3	2

Gambar 3.5 Contoh *Offspring* Bagian Penugasan Mesin pada Proses *Crossover* Dua Titik

7. Mutasi

Proses mutasi bertujuan untuk memperoleh individu baru sebagai kandidat solusi pada generasi berikutnya. Pada penelitian ini, proses mutasi dilakukan dengan menggunakan metode *swap mutation*. Proses mutasi dengan metode *swap mutation* dilakukan pada masing-masing bagian yaitu, pada bagian urutan operasi dan bagian penugasan mesin. Langkah-langkah *swap mutation* pada bagian urutan operasi antara lain:

a) Ambil *offspring* hasil *crossover* yang akan mengalami mutasi dengan membangkitkan terlebih dahulu bilangan acak antara 0 dan 1 untuk setiap *offspring*. Jika bilangan acak yang dihasilkan lebih kecil dari *mutation rate* (M_r) maka *offspring* tersebut akan menjalani proses mutasi. Contoh kromosom yang akan dimutasi dapat dilihat pada Gambar 3.6.

Induk (OS)	2	1	1	2	3	1

Gambar 3.6 Contoh Kromosom Induk pada Proses Mutasi

b) Pilih dua titik secara acak pada kromosom, gen yang terpilih akan diberi warna merah dan biru. Contoh dua titik acak pada kromosom dapat dilihat pada Gambar 3.7.

Induk (OS)	2	1	1	2	3	1

Gambar 3.7 Contoh Titik Acak Kromosom Induk pada Proses Mutasi

c) Tukar dua titik tersebut sehingga menghasilkan kromosom baru (*offspring*). Hasil kromosom baru dari penukaran dua titik tersebut dapat dilihat pada Gambar 3.8.

Offspring (OS)	2	3	1	2	1	1

Gambar 3.8 Contoh *Offspring* pada Proses Mutasi

Sementara itu, pada bagian penugasan mesin dapat diterapkan langkah-langkah *swap mutation* yang sama, sehingga hasil akhirnya berupa satu individu yang terdiri dari dua bagian hasil mutasi yaitu, bagian urutan operasi dan bagian penugasan mesin. Kemudian dilakukan pengecekan untuk memastikan bahwa setiap operasi dikerjakan oleh mesin yang tersedia. Apabila terdapat operasi yang dikerjakan oleh mesin yang tidak bersesuaian, ubah mesin tersebut dengan mesin yang tersedia pada operasi tersebut.

Setelah proses mutasi dilakukan, hitung nilai *fitness* dari masing-masing individu baru. Urutkan nilai *fitness* individu baru dan individu pada populasi generasi sekarang dari yang tertinggi ke yang terendah, lalu pilih individu terbaik pada generasi sekarang sebanyak jumlah *pop size* untuk digunakan pada generasi selanjutnya. Lakukan proses algoritma genetika hingga tercapai kriteria pemberhentian algoritma genetika, yaitu ketika generasi maksimum tercapai atau diperoleh nilai *fitness* = 1, sehingga diperoleh solusi terbaik dari algoritma genetika.

8. *Simulated Annnealing*

Setelah melakukan pencarian solusi dengan algoritma genetika untuk memperoleh solusi terbaik, langkah selanjutnya adalah melakukan pencarian solusi dengan menggunakan *simulated annealing*. Solusi terbaik dari algoritma genetika digunakan sebagai solusi awal pada pencarian solusi dengan *simulated annealing.* Penggunaan metode *simulated annealing* tidak hanya memberikan kesempatan untuk menerima solusi yang lebih baik, tetapi juga memberikan kesempatan untuk menerima solusi buruk yang memenuhi kriteria probabilitas penerimaan. Langkah-langkah pencarian solusi dengan menggunakan *simulated annealing* adalah sebagai berikut.

a. Tentukan solusi tetangga dengan memodifikasi solusi terbaik dari algoritma genetika. Modifikasi dilakukan dengan memilih dua titik secara acak pada bagian penugasan mesin, lalu tukar posisi kedua titik tersebut sehingga diperoleh solusi baru.
b. Cek apakah solusi tetangga telah valid (setiap operasi diproses pada mesin yang bersesuaian) dan memenuhi kendala-kendala yang ada. Jika solusi

tetangga tidak valid dan tidak memenuhi kendala-kendala yang ada maka lakukan kembali langkah 1. Jika sudah valid dan memenuhi kendala-kendala yang ada maka lanjut pada langkah 3.

c. Hitung nilai fungsi objektif dari solusi tetangga dan solusi terbaik dari algoritma genetika.

d. Hitung selisih nilai fungsi objektif antara solusi tetangga dengan solusi terbaik dari algoritma genetika.

   a. Jika selisih nilai fungsi objektif solusi tetangga dengan solusi terbaik dari algoritma genetika kurang dari nol, maka gantikan solusi terbaik algoritma genetika dengan solusi tetangga sebagai solusi.

   b. Jika selisih nilai fungsi objektif solusi tetangga dengan solusi terbaik algoritma genetika lebih dari nol, maka gantikan solusi terbaik dari populasi dengan solusi tetangga dengan ketentuan $\exp\left(-\frac{|\Delta E|}{T_i}\right) > r$

   di mana:

   - $\Delta E =$ selisih nilai fungsi objektif solusi tetangga dengan solusi terbaik algoritma genetika
   - $r =$ bilangan acak antara 0 sampai 1

   Kemudian, simpan solusi terbaik dari algoritma genetika sebagai solusi global.

e. Lakukan penurunan suhu dengan rumus:

$$T_{i+1} = T_i \times \alpha$$

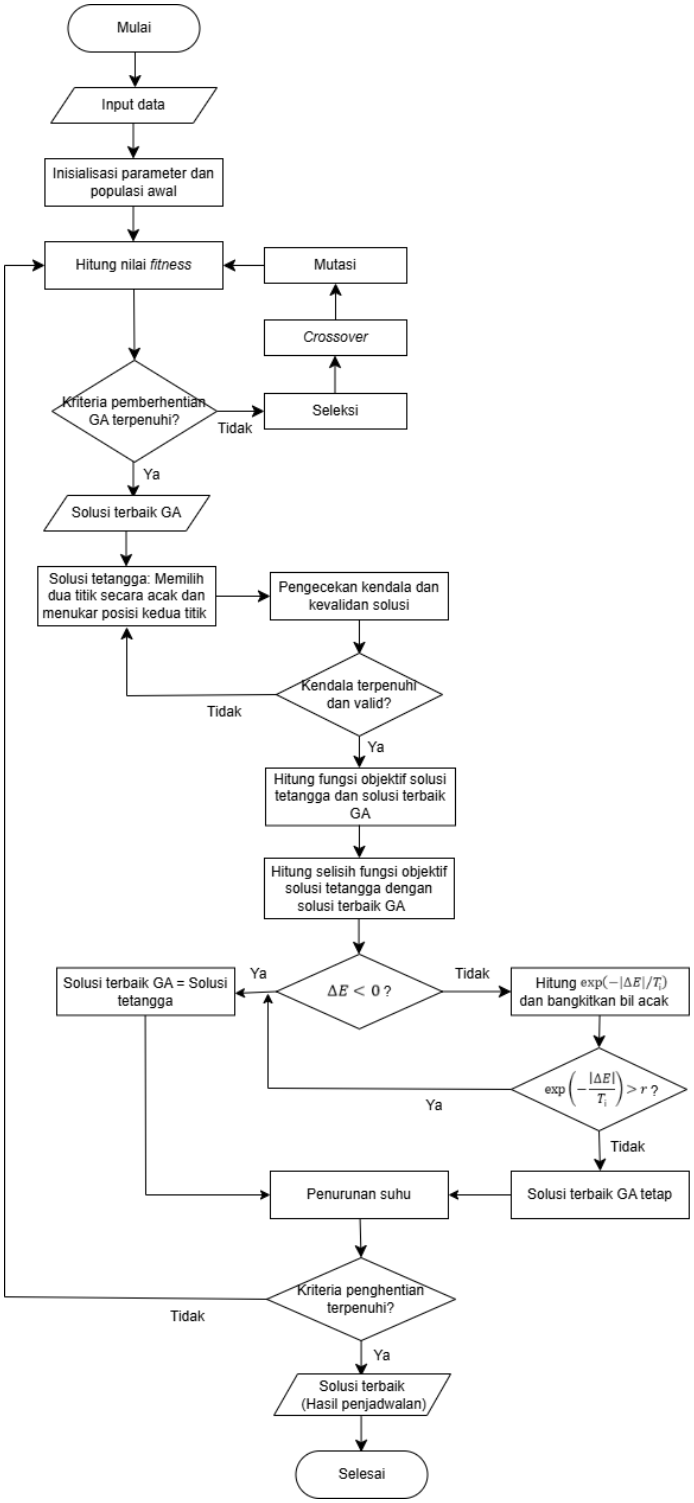
f. Jika nilai $T_{i+1} < T_f$ berhenti, namun jika $T_{i+1} \geq T_f$ ulangi kembali langkah-langkah dari pencarian solusi menggunakan algoritma genetika. Populasi yang digunakan adalah populasi terakhir algoritma genetika dengan memperhatikan hasil perbandingan dari solusi hasil modifikasi pada proses *simulated annealing* dengan solusi terbaik algoritma genetika.

9. Kriteria Penghentian Algoritma

   Pencarian dengan GASA akan dihentikan ketika suhu akhir (T_f) telah tercapai atau ketika nilai fungsi objektif telah mencapai 0. Solusi akhir atau solusi optimum diambil dari solusi global yaitu solusi dengan nilai

fungsi objektif terkecil.

Untuk memberikan gambaran terkait langkah-langkah GASA berikut disajikan *flowchart* yang dapat dilihat pada Gambar 3.9.



Gambar 3.9 *Flowchart* GASA