BAB III
MODEL CVRP DAN PENYELESAIANNYA DENGAN MENGGUNAKAN GABUNGAN ALGORITMA GENETIKA DAN SIMULATED ANNEALING

Bab ini berisi tentang *Capacitated Vehicle Routing Problem* (CVRP), model CVRP, dan langkah kerja gabungan algoritma *Genetic Algorithm* dan *Simulated Annealing* (GASA) dalam menyelesaikan CVRP.

3.1 *Capacitated Vehicle Routing Problem* (CVRP)

Diberikan sejumlah kendaraan dengan batasan kapasitas. Terdapat sejumlah pelanggan dengan permintaan tertentu untuk dilayani. *Capacitated Vehicle Routing Problem* (CVRP) adalah masalah pendistribusian sejumlah barang oleh kendaraan yang tersedia dengan kapasitas tertentu dari suatu depot ke sejumlah pelanggan lalu kembali ke depot. Jadi, CVRP adalah perluasan dari VRP dengan faktor tambahan bahwa setiap kendaraan punya kapasitas tersendiri. Setiap kendaraan melakukan pendistribusian sebanyak satu kali pengiriman yaitu dari depot ke setiap wilayah pelayanan lalu kembali ke depot.

Penyelesaian CVRP bertujuan untuk menentukan rute dengan total jarak terpendek. CVRP termasuk ke dalam masalah *NP-Hard Problem,* artinya usaha komputasi yang digunakan semakin meningkat seiring dengan meningkatnya ruang lingkup masalah. Akibatnya, jika solusi optimal dihitung dengan menggunakan metode eksak, maka diperlukan waktu komputasi yang lama. Oleh karena itu untuk menyelesaikan CVRP diperlukan metode metaheuristik untuk mengaproksimasi solusi optimum agar solusinya dapat diperoleh dengan cepat dan cukup baik.

3.2 Model CVRP

Pada bagian ini diturunkan model optimisasi dari CVRP. Adapun asumsi-asumsi yang diambil pada pemodelan ini adalah sebagai berikut:

1) Setiap pelanggan hanya dikunjungi tepat satu kali oleh satu kendaraan.
2) Setiap kendaraan mempunyai batasan kapasitas yang sama.

3) Setiap pelanggan terhubung satu sama lain dan jarak antara pelanggan i ke j sama dengan jarak pelanggan j ke i.
4) Kendaraan yang tersedia cukup untuk mengirim semua permintaan pelanggan.

Untuk menurunkan model CVRP, terlebih dahulu didefinisikan himpunan-himpunan yang digunakan oleh model sebagai berikut:

S : himpunan pelanggan

D : himpunan depot

K : himpunan kendaraan

Q : kapasitas kendaraan

d_i : permintaan pelanggan ke i

c_{ij} : jarak dari i ke j, dimana $i,j \in S \cup D$

Adapun variabel keputusan pada model didefinisikan sebagai berikut:

$$x_{ij}^k = \begin{cases} 1, \text{jika terdapat perjalanan dari i ke j dengan kendaraan k} \\ 0, \text{yang lainnya} \end{cases}$$

$$y_i^k = \begin{cases} 1, \text{jika kendaraan k mengunjungi pelanggan i} \\ 0, \text{yang lainnya} \end{cases}$$

Pada permasalahan CVRP akan dicari rute dengan total jarak terpendek untuk melakukan pendistribusian barang dari depot ke pelanggan-pelanggan lalu kembali ke depot dengan menggunakan sejumlah k kendaraan sehingga permintaan setiap pelanggan terpenuhi. Oleh karena itu fungsi tujuan dari CVRP dapat merepresentasikan sebagai berikut:

$$min\ Z = \sum_{k \epsilon K} \sum_{j \in S} \sum_{i \in S} c_{ij} x_{ij}^k$$

Adapun kendala-kendala dari model CVRP adalah sebagai berikut:

1) Setiap pelanggan hanya dikunjungi oleh 1 kendaraan, akibatnya kendala ini merepresentasikan oleh persamaan:

$$\sum_{k \in K} y_i^k = 1,\ \forall i \in S$$

2) Jumlah kendaraan yang keluar dan masuk ke depot ada sebanyak K. Akibatnya kendala ini merepresentasikan oleh persamaan:

$$\sum_{j\in S} x_{0j} = \sum_{j\in S} x_{j0} = K, 0 \in \mathrm{D}$$

3) Setiap kendaraan mengunjungi pelanggan tepat satu kali. Setelah itu kendaraan meninggalkan pelanggan tersebut dan melanjutkan perjalanan menuju pelanggan selanjutnya atau kembali menuju depot jika pelanggan tersebut adalah pelanggan terakhir dari sebuah rute. Ini berarti, kendaraan yang menuju pelanggan j akan meninggalkan pelanggan j. Akibatnya kendala ini merepresentasikan sebagai berikut:

$$\sum_{j\in S} x_{ij}^{k} = \sum_{j\in S} x_{jg}^{k} = y_i^k, \forall i, g \in S \cup D\ , \forall k \in K$$

4) Setiap kendaraan memiliki batasan kapasitas Q, sehingga jumlah permintaan pelanggan yang harus dipenuhi dalam satu rute tidak melebihi kapasitas maksimum kendaraan. Akibatnya kendala ini merepresentasikan sebagai berikut:

$$\sum_{i\in S} d_i y_i^k \leq Q, \forall k \in K$$

Pada sub bab selanjutnya akan dibahas mengenai teknik penyelesaian CVRP di atas dengan menggunakan gabungan Algoritma Genetika dan *Simulated Annealing.*

3.3 Algoritma GASA

Pada penelitian ini, CVRP akan diselesaikan dengan Algoritma GASA, yaitu gabungan antara Algoritma Genetika (GA) dan *Simulated Annealing* (SA). Algoritma GASA dikembangkan oleh Xun Gui Li pada 2008 untuk masalah optimasi pada reservoir berganda, serta digunakan oleh Tamilasari (2010) untuk masalah penjadwalan. Selanjutnya, Algoritma tersebut diaplikasikan oleh Sulistyo (2015) pada masalah *Multiple Depot Vehicle Routing Problem.* Gabungan dari algoritma GA dan SA dibangun untuk menutupi kekurangan dari algoritma GA dan SA. Algoritma GA dan SA memiliki kekurangan tersendiri. Algoritma GA memiliki kekurangan yaitu solusi yang diperoleh terindikasi bersifat konvergen prematur. Sedangkan SA memiliki kekurangan yaitu SA hanya dapat menyimpan

satu solusi terbaik setiap iterasinya dan mengabaikan solusi lainnya yang memungkinkan menghasilkan nilai yang lebih baik.

Algoritma GASA dirancang untuk menutupi kekurangan algoritma genetika dan *simulated annealing* sehingga diharapkan solusi yang dihasilkan adalah optimum global. GASA menerapkan operator-operator evolusi seperti *crossover*, mutasi, dan seleksi individu, serta variabel Temperatur digunakan untuk menentukan jumlah iterasi pada algoritma ini. Model CVRP pada sub bab sebelumnya akan diselesaikan dengan algoritma GASA (*Genetic Algorithm Simulated Annealing*).

Sebelum memulai algoritma GASA terlebih dahulu ditentukan nilai dari parameter-parameter yang diperlukan yaitu: temperatur (T), penurunan temperatur setiap iterasi (A), kriteria penghentian (Ɛ), banyak generasi, *population rate* (p), *crossover rate (cr)*, dan *mutation rate* (*mr*). Secara garis besar proses dalam mencari solusi yang optimum pada algoritma GASA dimulai dengan melakukan proses pencarian solusi oleh algoritma GA pada solusi awal sehingga diperoleh solusi terbaik dari GA. Setelah solusi terbaik GA didapatkan dilanjutkan ke proses pencarian oleh SA dengan memodifikasi solusi terbaik pada GA. Proses ini akan diulang sampai iterasi maksimum tercapai.

Untuk lebih memahami cara kerja algoritma GASA, berikut diberikan contoh kasus CVRP. Perusahaan MNO harus mengirimkan barang ke 10 pelanggan di tempat berbeda dengan 5 kendaraan dengan kapasitas maksimal 70 paket. Banyaknya permintaan dan posisi setiap pelanggan dinyatakan dalam koordinat garis lintang (x) dan garis bujur (y). Koordinat depot, koordinat pelanggan serta permintaan pelanggan dituliskan pada Tabel 3.1.

Tabel 3. 1
Koordinat Depot dan Pelanggan, serta Permintaan Setiap Pelanggan

Lokasi	X	Y	Permintaan
depot1	-6.9489	107.634	-
Pelanggan A	-6.9862	107.628	9
Pelanggan B	-6.9538	107.628	9
Pelanggan C	-6.9131	107.586	17
Pelanggan D	-6.877	107.618	11
Pelanggan E	-6.8959	107.588	17
Pelanggan F	-6.8812	107.583	16
Pelanggan G	-6.9018	107.582	12
Pelanggan H	-6.9018	107.613	12
Pelanggan I	-6.9672	107.575	20
Pelanggan J	-6.8947	107.597	17

Untuk menyelesaikan CVRP pada contoh di atas dilakukan langkah-langkah sebagai berikut:

1) Inisialisasi nilai T, A, dan Ɛ untuk proses SA. Untuk kasus di atas, diambil T=100, A = 0.8, dan Ɛ = 0.01
2) Inisialisasi banyak generasi yang dipilih dan jumlah kromosom dalam satu populasi. Misal diambil banyak generasi adalah 30 dan jumlah kromosom dalam satu populasi atau *population rate* adalah 10.
3) Merepresentasikan kromosom dalam bentuk *permutation encoding*. Kromosom dari *permutation encoding* adalah kromosom yang dibentuk dari bilangan *integer* yang merepresentasikan suatu urutan dalam satu rute. Setiap kromosom menyatakan sebuah rute yang dilewati oleh kendaraan. Misal pelanggan dinotasikan dengan 1,2,3,…,10. Salah satu contoh kromosom dari permasalahan di atas adalah kromosomy= 1 5 2 3 9 7 8 4 6 10.

   Kromosom di atas menyatakan urutan kendaraan yang mengunjungi pelanggan dari depot. Kendaraan memiliki batas kapasitas serta setiap pelanggan memiliki jumlah permintaan barang. Akibatnya jika jumlah barang di kendaraan tidak mencukupi untuk memenuhi permintaan pelanggan ke i di dalam rute

(kromosom) maka kendaraan akan kembali ke depot dan dikirim kendaraan lain untuk memenuhi permintaan pelanggan ke i dari depot.

4) Inisialisasi populasi awal sebanyak N kromosom.

Pada contoh kasus halaman sebelumnya, dibangkitkan 10 kromosom sebagai populasi awal. Misal kromosom tersebut adalah sebagai berikut:

kromosom1 = 8 6 2 4 9 7 3 10 5 1

kromosom2 = 4 10 5 3 9 6 8 1 7 2

kromosom3 = 6 3 5 9 8 10 7 1 4 2

kromosom4 = 3 4 7 9 10 8 2 6 5 1

kromosom5 = 3 9 2 8 7 4 6 5 1 10

kromosom6 = 4 1 10 7 6 8 3 9 5 2

kromosom7 = 3 5 4 9 6 7 10 8 1 2

kromosom8 = 10 7 5 9 1 6 8 2 3 4

kromosom9 = 5 7 6 2 8 4 3 9 10 1

kromosom10 = 7 4 1 2 9 10 5 8 6 3

5) Hitung nilai *fitness* dari masing-masing kromosom. Nilai *fitness* adalah nilai yang menyatakan baik atau tidaknya suatu individu untuk menjadi solusi. Nilai *fitness* ini dijadikan acuan dalam mencapai nilai optimal. Nilai *fitness* merepresentasikan kualitas sebuah kromosom sebagai solusi. Sehingga semakin kecil jarak yang dihasilkan nilai *fitness*-nya semakin besar dan sebaliknya. Fungsi *fitness* yang digunakan adalah

$$f = \frac{1}{z}$$

dengan z adalah total jarak suatu rute.

Misal koordinat node i adalah (x_1, y_1) dan koordinat node j adalah (x_2, y_2). Maka jarak antara i dan j dapat dihitung dengan menggunakan metode *haversine* sebagai berikut:

$$d = R \times 2 \times \sin^{-1}\left(\sqrt{\left(\sin\left(\frac{\Delta \text{lat}}{2}\right)\right)^2 + \cos(x_1) \times \cos(x_2) \times \left(\sin\left(\frac{\Delta \text{long}}{2}\right)\right)^2}\right)$$

dengan:

d= jarak *node* i ke j

R= jari-jari bumi (6371 km)

Δlat = $x_2 - x_1$

Δlong = $y_2 - y_1$

Nilai *fitness* populasi awal pada contoh diatas dituliskan pada Tabel 3.2

6) Pilih sebanyak m kromosom dari N jumlah kromosom dimana m adalah bilangan positif yang lebih kecil dari N. Kromosom tersebut dipilih melalui seleksi dengan metode *rank*. Seleksi *rank* merupakan suatu metode seleksi yang dilakukan dengan memberi urutan pada populasi sesuai dengan nilai *fitness*-nya, nilai *fitness* terkecil diberi nilai 1, yang terkecil kedua diberi nilai 2, dan seterusnya sampai yang terakhir diberi nilai N (jumlah kromosom dalam populasi).

Pada contoh kasus ini, dipilih m sebanyak 3 kromosom. Rank untuk populasi awal pada contoh di atas dituliskan pada Tabel 3.3. Berdasarkan Tabel 3.3 kromosom yang lolos seleksi dan masuk ke proses *crossover* adalah kromosom7, kromosom2, kromosom1.

Tabel 3.2
Jarak dan Nilai *Fitness* Populasi Awal

Rute	Jarak	*Fitness*
kromosom1	72.4948	0.01379
kromosom2	70.8606	0.01411
kromosom3	74.6924	0.01339
kromosom4	87.8487	0.01138
kromosom5	78.6674	0.01271
kromosom6	77.2444	0.01295
kromosom7	61.239	0.01633
kromosom8	81.2245	0.01231
kromosom9	75.9979	0.01316
kromosom10	75.3556	0.01327

Tabel 3.3
Urutan Kromosom Berdasarkan *Fitness*

Rute	*Fitness*	Rank
kromosom7	0.01633	1
kromosom2	0.01411	2
kromosom1	0.01379	3
kromosom3	0.01339	4
kromosom10	0.01327	5
kromosom9	0.01316	6
kromosom6	0.01295	7
kromosom5	0.01271	8
kromosom8	0.01231	9
kromosom4	0.01138	10

7) Sebelum memulai *crossover* pertama dilakukan pemilihan pasangan dari kromosom yang telah lolos proses seleksi secara acak. *Crossover* adalah proses untuk menyilangkan dua kromosom sehingga membentuk kromosom baru yang harapannya lebih baik dari pada induknya. Proses *crossover* dilakukan pada kromosom yang telah lolos dalam seleksi metode rank dengan metode *order crossover*. Pada *crossover* terdapat *crossover rate*(cr), nilai cr antara 0 sampai 1. Misal cr yang digunakan pada contoh di atas adalah 0.65, kemudian pilih sebuah bilangan acak antara 0 sampai 1. Jika bilangan acak yang dipilih kurang dari cr maka dilakukan *crossover*. Jika bilangan acak yang dipilih lebih dari cr maka tidak terjadi *crossover.* Langkah-langkah proses *order crossover* adalah sebagai berikut:
   a. Tentukan titik *crossover* dengan cara memilih dua nilai *integer* secara acak dari 1 sampai *n* banyaknya gen (jumlah pelanggan).
   b. Selanjutnya, buat kromosom kosong sebagai anak1 dan anak2 kemudian tempatkan gen-gen yang berada pada posisi diantara dua nilai *integer* yang telah didapat dari induk ke anak dengan posisi yang sama. Jadi, induk1 akan mewariskan gen-gen tersebut ke anak2 dan induk2 akan mewariskan gen-gen ke anak1.

c. Kemudian posisi gen yang kosong pada anak1 akan diisi dengan menempatkan urutan gen di induk1 yang belum termuat dan untuk posisi gen yang kosong pada anak2 akan diisi dengan menggunakan urutan gen di induk2 yang belum termuat.
d. Setelah seluruh gen terisi, maka anak 1 dan anak 2 merupakan kromosom baru hasil *crossover*.

Pada contoh kasus ini, diperoleh 3 kromosom yang akan melakukan *crossover* diantaranya kromosom7 dengan kromosom2, kromosom2 dengan kromosom1, dan kromosom7 dengan kromosom1. Proses *crossover* pada pasangan kromosom7 dengan kromosom2. Misal untuk pasangan ini diperoleh bilangan acak yaitu 0.34, akibatnya dilakukan crossover dengan metode *order crossover* adalah sebagai berikut:

1. Menentukan titik *crossover* pada induk. misal diperoleh dua bilangan *integer* acak yaitu 4 dan 6.

   Kromosom7 = 3 5 4 | 9 6 7 | 10 8 1 2

   kromosom2 = 4 10 5 | 3 9 6 | 8 1 7 2

2. Mewariskan gen diantara dua titik *crossover* pada kromosom7 ke anak2 juga pada kromosom2 ke anak1.

   anak1 = ---|3 9 6 |----

   anak2 = ---| 9 6 7 |----

3. isi ruang kosong gen anak1 dengan urutan gen pada kromosom7 yang belum termuat dan ruang kosong gen anak2 dengan urutan gen kromosom2 yang belum termuat

   anak1= 5 4 7 | 3 9 6 | 10 8 1 2

   anak2= 4 10 5 | 3 9 6 | 8 1 7 2

   kromosom baru hasil *crossover* antara kromosom7 dengan kromosom2 adalah

   anak1= 5 4 7 3 9 6 10 8 1 2

   dan

   anak2= 4 10 5 3 9 6 8 1 7 2

Proses *crossover* pada pasangan kromosom2 dengan kromosom1. Misal untuk pasangan ini diperoleh bilangan acak yaitu 0.66, akibatnya tidak dilakukan crossover. Sehingga gen-gen dari kromosom2 dan kromosom1 akan langsung diturunkan ke anak.

kromosom1 = 8 6 2 4 9 7 3 10 5 1

kromosom2 = 4 10 5 3 9 6 8 1 7 2

anak3 = 8 6 2 4 9 7 3 10 5 1

anak4 = 4 10 5 3 9 6 8 1 7 2

Proses *crossover* pada pasangan kromosom7 dengan kromosom1. Misal untuk pasangan ini diperoleh bilangan acak yaitu 0.51, akibatnya dilakukan crossover dengan metode *order crossover*:

1. Menentukan titik *crossover* pada induk. misal diperoleh dua bilangan *integer* acak yaitu 4 dan 6.
   kromosom7 = 3 5 4 | 9 6 7 | 10 8 1 2
   kromosom1 = 8 6 2 | 4 9 7 | 3 10 5 1
2. Mewariskan gen diantara dua titik *crossover* pada kromosom3 ke anak6 juga pada kromosom9 ke anak5.
   anak5 = ---|4 9 7|----
   anak6 = ---| 9 6 7 |----
3. isi ruang kosong gen anak5 dengan urutan gen pada kromosom7 yang belum termuat dan ruang kosong gen anak6 dengan urutan gen kromosom1 yang belum termuat
   anak5 = 3 5 6 | 4 9 7 | 10 8 1 2
   anak6 = 8 2 4 | 9 6 7 | 3 10 5 1

   kromosom baru hasil *crossover* antara kromosom7 dengan kromosom1 adalah

   anak5 = 3 5 6 4 9 7 10 8 1 2

dan

anak6 = 8 2 4 9 6 7 3 10 5 1

8) Lakukan proses mutasi pada *offspring* (kromosom anak) hasil *crossover* dan m kromosom terbaik pada generasi sekarang dengan probabilitas *mr*. Pada penelitian ini, metode mutasi yang digunakan adalah *swapping mutation. Swapping mutation* dilakukan dengan cara memilih dua nilai *integer* secara acak, lalu posisi gen pada dua bilangan *integer* tersebut ditukar. Misalkan dipilih mr = 0.55, kemudian memilih sebuah bilangan acak antara 0 sampai 1. Jika bilangan acak yang dipilih kurang dari *mr* maka terjadi mutasi. Jika bilangan acak yang dipilih lebih dari mr maka tidak terjadi mutasi.

Untuk kromosom anak1 hasil *crossover* diperoleh bilangan acak yaitu 0.32 akibatnya terjadi mutasi pada kromosom anak1. Misalkan dua bilangan acak yang diperoleh adalah 2 dan 5 maka gen pada posisi dua ditukar dengan gen pada posisi lima.

anak1= 5 **4** 7 3 **9** 6 10 8 1 2 => 5 9 7 3 4 6 10 8 1 2

anak1= 5 9 7 3 4 6 10 8 1 2

Untuk kromosom anak2 hasil *crossover* diperoleh bilangan acak yaitu 0.52 akibatnya terjadi mutasi pada kromosom anak2. Misalkan dua bilangan acak yang diperoleh adalah 3 dan 1, maka gen pada posisi tiga ditukar dengan gen pada posisi satu.

anak2= **4** 10 **5** 3 9 6 8 1 7 2 => 5 10 4 3 9 6 8 1 7 2

anak2= 5 10 4 3 9 6 8 1 7 2

Untuk kromosom anak3 hasil *crossover* diperoleh bilangan acak yaitu 0.22 maka terjadi mutasi pada kromosom anak3. Misalkan dua bilangan acak yang diperoleh adalah 10 dan 1, maka gen pada posisi sepuluh ditukar dengan gen pada posisi satu.

anak3 = **8** 6 2 4 9 7 3 10 5 **1** => 1 6 2 4 9 7 3 10 5 8

anak3 = 1 6 2 4 9 7 3 10 5 8

Untuk kromosom anak4 hasil *crossover* diperoleh bilangan acak yaitu 0.23 akibatnya terjadi mutasi pada kromosom anak4. Misalkan dua bilangan acak yang diperoleh adalah 4 dan 7, maka gen pada posisi empat ditukar dengan gen pada posisi tujuh.

anak4 = 4 10 5 **3** 9 6 **8** 1 7 2 => 4 10 5 **8** 9 6 **3** 1 7 2

anak4= 4 10 5 8 9 6 3 1 7 2

Untuk kromosom anak5 hasil *crossover* diperoleh bilangan acak yaitu 0.88 maka tidak terjadi mutasi pada kromosom anak5.

anak5 = 3 5 6 4 9 7 10 8 1 2

Untuk kromosom anak6 hasil *crossover* diperoleh bilangan acak yaitu 0.71 maka tidak terjadi mutasi pada kromosom anak6.

anak6 = 8 2 4 9 6 7 3 10 5 1

Untuk kromosom1 diperoleh bilangan acak yaitu 0.49 maka terjadi mutasi pada kromosom kromosom1. Misalkan dua bilangan acak yang diperoleh adalah 1 dan 2, maka gen pada posisi satu ditukar dengan gen pada posisi dua.

kromosom1 = **8 6** 2 4 9 7 3 10 5 1 => 6 8 2 4 9 7 3 10 5 1

Untuk kromosom2 diperoleh bilangan acak yaitu 0.92 maka tidak terjadi mutasi pada kromosom kromosom2.

kromosom2 = 4 10 5 3 9 6 8 1 7 2

Untuk kromosom7 diperoleh bilangan acak yaitu 0.69 maka tidak terjadi mutasi pada kromosom kromosom7.

Kromosom7 = 3 5 4 9 6 7 10 8 1 2

9) Hitung nilai *fitness* dari masing-masing kromosom baru seperti pada tahap ke 5. Pada contoh di atas, diperoleh nilai *fitness* pada kromosom baru dituliskan pada Tabel 3.4.
10) Urutkan *fitness* dari kromosom baru dan kromosom pada populasi generasi sekarang, lalu pilih N kromosom terbaik pada generasi sekarang untuk menuju generasi selanjutnya.

    Pada contoh di atas dipilih 10 kromosom untuk masuk ke generasi selanjutnya. Urutan kromosom baru dan kromosom pada populasi generasi sekarang dituliskan pada Tabel 3.5.

    Berdasarkan Tabel 3.5 generasi selanjutnya didapat populasi baru yaitu: kromosom7, anak4, kromosom2, kromosom3, anak6, anak1, kromosom1, anak3, kromosom10, dan kromosom9.

Tabel 3.4
Fitness dari Kromosom Baru

Rute	Jarak	*Fitness*
anak1	62.6618	0.015958676
anak2	73.8557	0.013539923
anak3	84.6052	0.011819604
anak4	70.6303	0.014158225
anak5	55.6486	0.017969919
anak6	79.1587	0.012632845
kromosom1	74.57478	0.013409359
kromosom2	70.8606	0.014112216
kromosom7	61.23896	0.016329475

Tabel 3.5
Urutan *Fitness* Kromosom Baru dengan Kromosom pada Populasi Awal

Rute	Jarak	*Fitness*
anak5	55.6486	0.017969919
kromosom7	61.239	0.01633
anak1	62.6618	0.01596
anak4	70.6303	0.01416
kromosom2	70.86059	0.014112
anak2	73.85567	0.01354
kromosom1	74.5748	0.01341
kromosom3	74.6924	0.01339
kromosom10	75.3556	0.01327
kromosom9	75.99791	0.013158
kromosom6	77.2444	0.01295
kromosom5	78.6674	0.01271
anak6	79.15873	0.012633
kromosom8	81.2245	0.012312
anak3	84.6052	0.01182
kromosom4	87.8487	0.01138

11) Ulangi langkah 5) sampai 10) dilanjutkan hingga mencapai generasi maksimum. Sehingga diperoleh solusi terbaik algoritma GA. Misal untuk contoh di atas diperoleh solusi terbaik GA adalah anak5 = 3 5 6 4 9 7 10 8 1 2.
12) Kemudian jalankan *Simulated Annealing* dengan cara sebagai berikut:
13) Jika T>Ɛ, maka

14) Modifikasi kromosom terbaik pada algoritma GA dengan cara dengan cara memilih dua nilai *integer* secara acak, lalu posisi gen pada dua bilangan *integer* tersebut ditukar. Sehingga diperoleh kromosom baru disebut sebagai kromosomx. Misalkan diperoleh dua bilangan *integer* acak yaitu 3 dan 5, akibatnya gen pada posisi tiga dan posisi lima pada anak5

    anak5 = 3 5 6 4 9 7 10 8 1 2

    dimodifikasi menjadi

    kromosomx = 3 5 9 4 6 7 10 8 1 2

15) Hitung *fitness* kromosom baru hasil modifikasi kromosom terbaik pada GA. Pada contoh di atas diperoleh total jarak dan nilai *fitness* pada kromosomx sebagai berikut:

Tabel 3.6
Jarak dan Nilai *Fitness* pada Kromosomx

Rute	Jarak	*Fitness*
Kromosomx	57.54237	0.01737

16) Bandingkan anak5 dengan Kromosomx.

a. Jika selisih *fitness* kromosom baru dikurangi kromosom terbaik dari GA lebih dari nol maka gantikan kromom terbaik GA dengan kromom baru sebagai solusi.

b. Jika selisih *fitness* kromom baru dengan kromosom kurang dari nol maka gantikan kromosom terbaik dari populasi dengan kromosom baru jika peluang $\exp(-|\Delta E|/T) > \delta$;

    dimana

    ΔE adalah selisih nilai *fitness* kromosom baru dikurangi kromosom terbaik

    δ adalah sebuah bilangan acak antara 0 sampai 1.

    Misalkan pada contoh ini nilai dari δ adalah 0.42

    Fitness kromosomx - *fitness* kromosom7 = 0.01737 - 0.01797 = -0.000591419

Pada contoh di atas, selisih antara kromosomx dengan kromosom7 kurang dari 0, sehingga dilakukan tahapa pada 16) bagian b

Nilai dari exp(-|ΔE|/T) = exp (-0.000591419 / 100) = 0.999994086

Akibatnya kromosomx menggantikan anak 5 pada populasi GA pada iterasi selanjutnya.

17) Kurangi nilai T, karena dipilih A = 0.8 akibatnya pada iterasi selanjutnya nilai T pada iterasi selanjutnya adalah 80

18) jika nilai T < Ɛ, stop. Jika T≥ Ɛ ulangi langkah 6) dengan populasi yang digunakan adalah populasi terakhir pada GA dengan memperhatikan hasil perbandingan dari kromosom baru hasil modifikasi pada proses SA dengan kromosom terbaik dari GA. Pada contoh diatas, jika T≥ Ɛ maka ulangi langkah 6) dengan populasi awal sebagai berikut:

Tabel 3.7
Populasi Awal untuk Proses GA Selanjutnya

Rute	Jarak	*Fitness*
Kromosomx	57.54237	0.017378
kromosom7	61.239	0.01633
anak1	62.6618	0.01596
anak4	70.6303	0.01416
kromosom2	70.86059	0.014112
Kromosom1	72.4948	0.01379
anak2	73.85567	0.01354
Kromosom3	74.3686	0.01345
kromosom10	75.3556	0.01327
kromosom9	75.99791	0.013158