BAB IV

PEMBAHASAN

Pada bab ini akan diberikan penjelasan mengenai penggunaan *hybrid* algoritma *Genetic Algorithms* (GA) dan *Ant Colony Optimization* (ACO) untuk menyelesaikan masalah *Dynamic Travelling Salesman Problem* (DTSP).

4.1. *Hybrid* Algoritma ACO dan GA

Hybrid algoritma *Genetic Algorithms* (GA) dan *Ant Colony Optimization* (ACO) adalah merupakan gabungan dari kedua algoritma tersebut dengan melakukan algoritma ACO terlebih dahulu kemudian diakhiri oleh proses algoritma genetika hingga mendapat solusi yang dicari. Penyelesaian masalah *Dynamic Travelling Salesman Problem* (DTSP) menggunakan *hybrid* algoritma ACO dan algoritma genetika diawali dengan membuat tabel pheromone awal dengan persamaan $\frac{1}{jumlah\_kota}$ kemudian mencari solusi *tabu list* melalui perhitungan probabilitas pilih kota dan solusi *tabu list* yang dihasilkan akan melalui proses seleksi dengan metode *roulette wheel* dalam algoritma genetika sehingga mendapatkan sebuah solusi jarak paling pendek. Kemudian solusi tersebut dapat diubah sesuai permasalahan yang ada dengan menggunakan algoritma ACO, setelah itu dilanjutkan kembali dengan algoritma genetika.

Proses *hybrid* algoritma ACO dan GA ini akan berhenti setelah mencapai $\max\_ite$. Prosedur *hybrid* algoritma ACO dan GA dijelaskan dalam Gambar 4.1.

Prosedur *hybrid* algoritma ACO dan GA

```
begin
    input data dan inisialisasi parameter();
    do
      generate pheromoneAwal();
          for i = 0 sampai ite
                    do
                      for  t = 1 sampai jml_kota
                      hitung nilaiProbPilihKota();
                      mengisi tabu();
                      hitung jarakTotal();
                      update pheromone();
          while (t = i+1) kosongkan tabu();
          for i = 0 sampai ite
                seleksi RouletteWheel();
                proses crossover();
                proses mutasi();
                update populasiBaru();
              while (i = ite) cetak jarakTotal_terpendek();
    perbarui jml_kota();
    while (ite);
end
```

Gambar 4.1. Prosedur *hybrid* algoritma ACO dan GA

Berikut ini adalah penjelasan secara rinci tentang prosedur penyelesaian *Dynamic Travelling Salesman Problem* (DTSP) menggunakan *hybrid* algoritma ACO dan algoritma genetika.

4.1.1. Input Data dan Inisialisasi Parameter

Prosedur ini merupakan prosedur untuk menginputkan data dan menentukan informasi awal yang dibutuhkan dalam *hybrid* algoritma ACO dan algoritma genetika. Data yang diinputkan merupakan data tabel jarak antar kota. Parameter-parameter yang diperlukan adalah *alpha* yang merupakan nilai pengaruh relatif feromon, *beta* yang merupakan pengaruh relatif kota terdekat (*visibility*), *pop_size* yang merupakan jumlah populasi semut yang nantinya akan menentukan banyaknya solusi *tabu list*, *jml_kota* yang merupakan jumlah

kota, ρ yang merupakan koefisien penguapan intensitas *pheromone*, *max_ite* yang merupakan banyak iterasi algoritma *hybrid* ACO GA, p_c merupakan probabilitas dari *crossover,* p_m merupakan probabilitas dari mutasi.

. Berikut adalah prosedur untuk pengisian parameter yang disajikan dalam Gambar 4.2.

Prosedur Input Data dan Inisialisasi Parameter

```
begin
    nilai α ← alpha;
    nilai β ← beta;
    jumlah semut ← pop_size;
    jumlah kota ← jml_kota;
    jumlah iterasi ← ite;
    nilai ρ ← rho;
    laju crossover ← pc;
    laju mutasi ← pm;
end
```

Gambar 4.2. Prosedur input data dan inisialisasi parameter

Jumlah dari *pop_size* merupakan jumlah dari solusi *tabu list* yang digunakan pada algoritma genetika.

4.1.2. Membuat Pheromone Awal

Membuat pheromone awal adalah prosedur membuat tabel pheromone dengan rumusan $= {}^{1}/_{jml\_kota}$, pheromone awal akan bernilai sama untuk setiap kota karena probabilitas awal pemilihan kota oleh semut (ketertarikan semut memilih kota) bernilai sama. Fungsi tujuannya adalah meminimalkan jarak *tour* dari *jml_kota*, *tour* dimulai dari satu kota awal terpilih secara *random*

dan melalui kota lainnya tepat satu kali serta kembali lagi ke kota awal. Prosedur untuk inisialisasi solusi awal ditunjukkan pada Gambar 4.3.

Prosedur Membuat Pheromone Awal

begin
for *i*=0 sampai *jml_kota*
for *j*=0 sampai *jml_kota*
if (*i* = *j*) phero = 0;
else
phero = $(\frac{1}{jml\_kota})$;
end

Gambar 4.3. Prosedur membuat pheromone awal

Setelah membuat tabel pheromone awal, kemudian semut akan memilih kota awal secara acak. Lalu dilanjutkan proses menghitung nilai probabilitas pilih kota untuk pemilihan kota berikutnya. Kota yang terpilih akan diisi ke dalam *tabu list.*

4.1.3. Menghitung Nilai Probabilitas Pilih Kota Dan Mengisi Tabu

Langkah ini merupakan prosedur untuk menghitung nilai probabilitas pilih kota dengan menggunakan persamaan **2.3**. Semut lebih menyukai titik yang dekat dan terhubung dengan tingkat pheromone yang tinggi. Untuk membangun jalan terpendek yang mungkin, setiap semut mempunyai suatu bentuk *memory* yang disebut *tabu list*. *Tabu list* digunakan untuk menentukan himpunan titik yang masih harus dikunjungi pada setiap langkah dan untuk menjamin terbentuknya jalan terpendek yang mungkin. *Tabu list* juga berfungsi untuk mencegah terjadinya kunjungan kekota yang sama lebih dari satu kali. Langkah perhitungan probabilitas pilih kota ini berjalan hingga *tabu list*

berisikan kode permutasi dari setiap kota dan jumlah *tabu list* bernilai sama dengan jumlah *pop_size*. Prosedur menghitung nilai probabilitas pilih kota disajikan pada Gambar 4.4.

Prosedur hitung nilai probabilitas pilih kota dan isi tabu

```
begin
    kotaawal = random(jml_kota);
    tabu_list[0] = kotaawal;
            for i=0 sampai pop_size
                    index = (random() * jml_kota);
                    tabu_list[i][0]=domain[index];
                    for i=0 sampai pop_size
                            peny = 0;
                            for j=0 sampai jml_kota
                                if (tabu_list[i][0] != domain[j])
                     jarak = 1 / jarak[tabu_list[i][0]][domain[j]]);
                     a = pow(phero[tabu_list[i][0]][domain[j]], alpha);
                     b = pow(jarak, beta);
                     peny = peny + (a * b);

                    penyebut[i] = peny;
                    for i=0 sampai pop_size
                    for j=0 sampai jml_kota
                            if (tabu_list[i][0] != domain[j])
                     a = pow(phero[tabu_list[i][0]][domain[j]], beta);
                     b = pow((float)1 / jarak[tabu_list[i][0]][domain[j]], alpha);
                     prob[i][j] = (a * b) / penyebut[i];
                              else
                    prob[i][j] = 0;
    for i=0 sampai pop_size
       for j=0 sampai jml_kota-1
            tabu_list[i][j];
end
```

Gambar 4.4. Prosedur hitung nilai probabilitas pilih kota dan isi tabu

Setelah menghitung nilai probabilitas pilih kota, kemudian dilanjutkan proses menghitung jarak total *tour* atau perjalanan. Perhitungan ini disesuaikan dengan urutan kode permutasi dalam tiap *tabu list.* Jarak total untuk setiap semut didapatkan dan *tabu list* dikosongkan kembali untuk digunakan lagi pada iterasi berikutnya.

4.1.4. Memperbarui Pheromone

Prosedur ini merupakan prosedur untuk membuat tabel pheromone yang baru. Setelah semua semut membangun sebuah *tour*, pheromone akan di-*update* dengan cara mengurangi tingkat pheromone oleh suatu konstanta dan kemudian semut meletakkan pheromone pada *edge* yang dilewati. *Update* dilakukan sedemikian rupa sehingga *edge* dari lintasan yang lebih pendek dan dilewati banyak semut menerima jumlah pheromone yang lebih banyak. Karena itu, pada iterasi berikutnya akan mempunyai probabilitas yang lebih tinggi untuk dipilih. Prosedur memperbarui pheromone ditunjukkan pada Gambar 4.5.

Prosedur memperbarui pheromone

```
begin
        for i = 0 sampai pop_size
                for r = 0 sampai jml_kota
                        if (r! = jml_kota-1)
                do
                 jml[i][tabu[i][r]][tabu[i][r+1]]=Q/jar_total[i];
                else
                 jml[i][tabu[i][0]][tabu[i][r+1]]=Q/jar_total[i];

        for j = 0 sampai jml_kota
                for r = 0 sampai jml_kota
        j1=0;
                for i=0 sampai pop_size
                        j1=j1+jml[i][j][r];
                        j1=j1+jml[i][r][j];

                c=(rho*phero[j][r])+j1;
                phero[j][r]=c;
end
```

Gambar 4.5. Prosedur memperbarui pheromone

Setelah prosedur memperbarui pheromone mencapai iterasi maksimal, akan dilakukan perhitungan jarak tempuh total semut dari rute yang dibangun.

4.1.5. Seleksi

Pada prosedur ini akan dicari fungsi *fitness* yang digunakan untuk mengukur tingkat kebaikan atau kesesuaian suatu solusi dengan solusi yang dicari. Sejumlah solusi yang dibangun oleh ACO akan dievaluasi menggunakan fungsi *fitness*. Prosedur *hybrid* ACO dan GA menggunakan seleksi *roulette wheel*. Proses seleksi *roulette whell* dapat dibayangkan dengan sebuah lingkaran roda lotere yang daerahnya dibagi-bagi sebanyak solusi yang dibangun. Semakin besar nilai *fitness*nya semakin diharapkan individu tersebut terpilih. Prosedur seleksi *roulette whell* ditunjukkan pada Gambar 4.6.

```
Prosedur seleksi

begin
    n = fitkum.length;
    calonInduk[]=new [n];
            for i = 0 sampai n
            random (0,1);
                    for j = 0 sampai j < n
        if (random<=fitkum[j])
        calonInduk[i] = j;
        break;

    return calonInduk;
end
```

Gambar 4.6. Prosedur seleksi

Proses seleksi ini dilakukan untuk memilih, mana anggota solusi yang bisa menjadi induk (*parent*) untuk proses kawin silang (*crossover*) dan mutasi.

4.1.6. Crossover

Crossover sering disebut kawin silang. *Crossover* dilakukan untuk mendapatkan harapan kombinasi yang lebih baik antara satu individu dengan individu yang lainnya. Parameter paling penting dalam *crossover* adalah laju *crossover*. Laju *crossover* cenderung memiliki nilai yang cukup tinggi. Laju crossover adalah nilai harapan induk akan mengalami kawin silang. Nilai laju *crossover* (p_c) direkomendasikan antara 0,8 sampai 0,95. Namun untuk beberapa persoalan *crossover*, $p_c = 0{,}6$ memiliki hasil yang paling baik. Perhitungan pada Gambar 4.7.

Prosedur seleksi crossover

```
begin
LinkedList induk=new LinkedList();
for i = 0 sampai data.length
    random (0,1);
    if (random <= Pc)
        induk.add(calonInduk[i]);
    if (induk.size()%2!=0)
        induk.removeLast();
end
```

Gambar 4.7. Prosedur seleksi crossover

Laju *crossover* yang semakin besar mengakibatkan meningkatnya jumlah kromosom terpilih yang melakukan *crossover* atau kawin silang. Prosedur untuk mendapatkan anak hasil crossover ditunjukkan pada Gambar 4.8.

Prosedur mencari anak crossover

```
begin
    n_anak=induk.size();
    anak[][]=new int[n_anak][data[0].length];
    pjg=data[0].length;
            for i = 0 sampai n_anak
                    random (1 sampai n_anak.length);
             for j = 0 sampai j < data[0].length
                    anak[i][j]=data[(int)induk.get(i)][j];
                    anak[i+1][j]=data[(int)induk.get(i+1)][j];

        for  j = random sampai j < pjg
           anak[i][j]=anak[i+1][j];
           anak[i+1][j]=swap;

        benarkan(random,anak[i],data[(int)induk.get(i)]);
        benarkan(random,anak[i+1],data[(int)induk.get(i+1)]);
              tampil(anak[i]);
              tampil(anak[i+1]);

     return anak;
    end
```

Gambar 4.8. Prosedur mencari anak crossover

Anak crossover adalah induk-induk baru hasil prosedur crossover yang akan diseleksi lagi dalam proses pemilihan induk baru dalam generasi berikutnya.

4.1.7. Mutasi

Mutasi memungkinkan kemunculan individu-individu baru yang bukan berasal dari hasil kawin silang atau sama sekali berbeda dengan individu yang sudah ada. Pada masalah ini, mutasi mengacu pada perubahan urutan atau penggantian elemen dari vektor solusi. Parameter penting saat prosedur mutasi adalah laju mutasi. Nilai laju mutasi cenderung kecil. Nilai laju mutasi akan menentukan kromosom mana yang akan mengalami perubahan gen. Prosedur mutasi ditunjukkan pada Gambar 4.9.

Prosedur mutasi

```
begin
LinkedList induk=new LinkedList();
for i = 0 sampai data.length
    random (0,1);
    if (random <= Pm)
        induk.add(calonInduk[i]);
end
```

Gambar 4.9. Prosedur seleksi mutasi

Laju mutasi yang semakin besar mengakibatkan meningkatnya jumlah kromosom terpilih yang melakukan mutasi atau berbeda. Prosedur untuk mendapatkan anak hasil mutasi ditunjukkan pada Gambar 4.10.

Prosedur mencari anak mutasi

```
begin
    anakMutasi[][]=new int[induk.size()][data[0].length];
    for i=0 sampai anakMutasi
       for j=0 sampai data[0].length
       rand1 = random*anakMutasi[0].length;
       rand2 = rand1;
       do
              rand2 = random*anakMutasi[0].length;
       while (rand2==rand1)
              for j=0 sampai anakMutasi[0].length
                     if (j==rand1)
                            anakMutasi[i][j]=data[(int)induk.get(i)][rand2];

                     else if (j==rand2)
                            anakMutasi[i][j]=data[(int)induk.get(i)][rand1];
                     else
                            anakMutasi[i][j]=data[(int)induk.get(i)][j];

    return anakMutasi;
end
```

Gambar 4.10. Prosedur mencari anak mutasi

Anak mutasi adalah induk-induk baru hasil prosedur mutasi yang akan diseleksi lagi dalam proses pemilihan induk baru dalam generasi berikutnya.

4.1.8. Memperbarui populasi

Prosedur ini merupakan tahapan akhir algoritma genetika. Setelah proses pencarian individu-individu baru hasil dari proses *crossover* dan mutasi, kemudian melakukan pengelompokan individu lama, individu baru hasil *crossover* dan individu baru hasil mutasi. Selanjutnya individu ini akan diseleksi lagi hingga mencapai *pop_size* sejumlah solusi awal pada tahap algoritma genetika dengan menggunakan ranking *ascending* jarak totalnya. Prosedur untuk memperbarui populasi ini dijelaskan pada Gambar 4.11.

```
Prosedur memperbarui populasi

begin
    solusiawal();
    solusianak crossover();
    solusianakmutasi();
            do
            gabung(solusiawal, solusianak crossover, solusianakmutasi);
    for i = 0 sampai pop_size
            ranking();
end
```

Gambar 4.11. Prosedur memperbarui populasi

Prosedur memperbarui populasi bertujuan untuk menjaga induk dengan bobot jarak kecil bisa dipertahankan dan masuk dalam iterasi berikutnya.

4.1.9. Menambah atau mengurangi kota tujuan

Prosedur ini merupakan prosedur menambah kota tujuan pada solusi terbaik atau mengurangi kota tujuan pada solusi terbaik hasil *hybrid* algoritma ACO dan GA. Jika penambahan kota atau pengurangan kota terjadi setelah kota *n*, maka akan dibentuk kembali *pheromone* awal dengan jumlah kota tujuan

sejumlah banyak kota tujuan setelah kota *n* dan kota awal adalah kota *n*. Prosedur menambah atau mengurangi kota tujuan ditunjukkan pada Gambar 4.12.

Prosedur tambah atau delete kota

```
begin
    inisialisasi update();
    membentuk Pheromone baru();
            for i = 0 sampai ite
                    ACO();
                    GA();
            Solusi terbaik();
    bentuk rute akhir()= rute update()+solusi sebelum update();
end
```

Gambar 4.12. Prosedur tambah atau hapus kota

Terdapat perubahan kota tujuan dengan hapus atau tambah kota dan memulai kembali algoritma *hybrid* ACO-GA untuk menyelesaikan permasalahan DTSP tersebut.

4.2. Data

Data-data yang digunakan merupakan data sekunder yang diambil dari dua sumber, yaitu sebagai berikut:

1. Data jarak antar kota yang digunakan diperoleh dari alamat website : www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/XML-TSPLIB/instances/swiss42.xml terdiri dari 42 kota. Data lengkap jarak antar kota swiss42 dapat dilihat pada **Lampiran 2**.
2. Data jarak antar kota yang digunakan dalam pembahasan ini adalah kroA150 yang diambil dari www.iwr.uni-heidelberg.de/ groups/ comopt/ software/ TSPLIB95/ XML-TSPLIB/ instances/ yang terdiri dari 150 kota. Data lengkap jarak antarkota kroA150 dapat dilihat pada **Lampiran 3**.

4.3. Penyelesaian Secara Manual Contoh Permasalahan Dynamic Travelling Salesman Problem Menggunakan Data 10 Kota

Misalkan dalam menyelesaikan persoalan DTSP pada data jarak 10 kota di Swiss42.xml menggunakan prosedur *hybrid Ant Colony Optimization* (ACO) dan *Genetic Algorithms* (GA) untuk mencari yang jaraknya minimal.

4.3.1. Input Data dan Inisialisasi Parameter

Pada langkah inisialisasi parameter, dipilih nilai sebagai berikut :

$\alpha = 1$, $\beta = 1$, $\rho = 0.5$, $pop\_size = 5$, $p_c = 0.5$, $p_m = 0.05$ dan $jml\_kota = 10$.

4.3.2. Membuat Pheromone Awal

Tabel pheromone awal dibuat dengan rumusan $\frac{1}{jml\_kota}$. Terdapat 10 kota yaitu kota A=kota 1, kota B=kota 2, kota C=kota 3, kota D=kota 4, kota E=kota 5, kota F=kota 6, kota G=kota 7, kota H=kota 8, kota I=kota 9, dan kota J=kota 10. Karena *jml_kota* = 10 maka nilai pheromone awal = $\frac{1}{10}$ = 0,1. Nilai keseluruhan dari pheromone awal ditunjukkan pada Tabel 4.1.

Tabel 4.1. Tabel pheromone awal

	A	B	C	D	E	F	G	H	I	J
A	0	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
B	0,1	0	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1
C	0,1	0,1	0	0,1	0,1	0,1	0,1	0,1	0,1	0,1
D	0,1	0,1	0,1	0	0,1	0,1	0,1	0,1	0,1	0,1
E	0,1	0,1	0,1	0,1	0	0,1	0,1	0,1	0,1	0,1
F	0,1	0,1	0,1	0,1	0,1	0	0,1	0,1	0,1	0,1
G	0,1	0,1	0,1	0,1	0,1	0,1	0	0,1	0,1	0,1
H	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0	0,1	0,1
I	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0	0,1
J	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0

4.3.3. Menghitung Nilai Probabilitas Pilih Kota Dan Mengisi Tabu

Setelah dihitung nilai pheromone awal, masing-masing semut akan memilih kota awal secara acak sebab semut pertama kali memilih jalan secara acak dan semut selanjutnya cenderung memilih rute yang telah terdapat pheromone dan semakin banyak tingkat pheromone maka semakin banyak semut yang melewati jalan tersebut. Didapatkan hasil pada Tabel 4.2 :

Tabel 4.2. Semut memilih secara acak

Semut	Kota
1	C
2	H
3	E
4	D
5	A

Berdasarkan kota awal yang terpilih tersebut kemudian diisi ke *tabu list. Tabu list* yang terbentuk akan seperti Tabel 4.3 :

Tabel 4.3. Tabulist awal

Semut	Tabulist
1	3
2	8
3	5
4	4
5	1

Kemudian masing-masing semut memulai perjalanannya ke kota berikutnya menggunakan persamaan **2.3**. Langkah ke-1 atau s=1, misal semut pertama kota awalnya adalah 3 atau kota C, maka dihitung :

$$\text{Penyebut} = \sum_{k=1}^{m} phero_{ij}^{\ \alpha} * \left(\frac{1}{matrik\_jarak_{ij}}\right)^{\beta} \quad \text{dengan syarat } j \notin \text{tabu}$$

$$= 0.1^1 * \left(\frac{1}{30}\right)^1 + 0.1^1 * \left(\frac{1}{34}\right)^1 + 0.1^1 * \left(\frac{1}{11}\right)^1 + 0.1^1 * \left(\frac{1}{18}\right)^1 + 0.1^1 * \left(\frac{1}{57}\right)^1$$

$$+ 0.1^1 * \left(\frac{1}{36}\right)^1 + 0.1^1 * \left(\frac{1}{65}\right)^1 + 0.1^1 * \left(\frac{1}{62}\right)^1 + 0.1^1 * \left(\frac{1}{84}\right)^1$$

$$= 0.003333 + 0.002941 + 0.009090 + 0.005555$$

$$+0.001754 + 0.002777 + 0.001538 + 0.001613 + 0.001190$$

$$= 0{,}029794979 .$$

Sehingga untuk kota 1 atau kota A diperoleh probabilitas dipilihnya adalah :

$$= \frac{phero_{ij}{}^{\alpha} * \left(\frac{1}{matrik_{jarak_{ij}}}\right)^{\beta}}{penyebut}$$

$$= \frac{phero_{31}{}^{1} * \left(\frac{1}{matrik_{jarak_{31}}}\right)^{1}}{penyebut}$$

$$= \frac{0.1^{1} * \left(\frac{1}{30}\right)^{1}}{0{,}029794979}$$

$$= \frac{0.003333}{0{,}029794979}$$

$$= 0{,}111875673$$

Demikian juga untuk kota 2 atau kota B diperoleh probabilitasnya adalah :

$$= \frac{phero_{ij}{}^{\alpha} * \left(\frac{1}{matrik\_jarak_{ij}}\right)^{\beta}}{penyebut}$$

$$= \frac{phero_{32}{}^{1} * \left(\frac{1}{matrik_{jarak_{32}}}\right)^{1}}{penyebut}$$

$$= \frac{0.1^1 * \left(\frac{1}{34}\right)^1}{0{,}029794979}$$

$$= \frac{0.002941}{0{,}029794979}$$

$$= 0{,}098713829$$

Proses ini berlanjut hingga kota ke-10 atau kota J. Untuk hasil lengkap nilai probabilitas pilih kota bisa dilihat pada Tabel 4.4 yaitu :

Tabel 4.4. Nilai probabilitas pilih kota s=1

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	3	0,112	0,099	0	0,305	0,186	0,059	0,093	0,052	0,054	0,04
2	8	0,164	0,19	0,093	0,113	0,105	0,11	0,135	0	0,049	0,041
3	5	0,094	0,111	0,167	0,273	0	0,075	0,15	0,052	0,045	0,033
4	4	0,117	0,117	0,245	0	0,245	0,056	0,103	0,05	0,038	0,029
5	1	0	0,247	0,124	0,161	0,116	0,067	0,112	0,1	0,04	0,033

Membangkitkan bilangan acak r_i dengan range [0,1] dengan i = 1,2,… , *pop_size*. Kemudian untuk semut 1, diperoleh bilangan acak = 0,616. Setelah itu, mencari kota dimana bilangan acak tersebut mendekati nilai jumlahan probabilitas pilih kota. Karena diperoleh bilangan acak = 0,616, maka semut akan memilih kota 5 atau kota E, sebab nilai jumlahan probabilitasnya = 0,112+0,099+0,305+0,186 = 0,702. Dengan cara yang sama dapat dibentuk tabel bilangan acak dan kota terpilih pada Tabel 4.5 :

Tabel 4.5. Pengisian tabulist saat s=1

Semut	Bilangan acak	Pilih kota	Tabu
1	0,616	5	3,5
2	0,060	1	8,1
3	0,643	4	5,4
4	0,404	3	4,3
5	0,304	3	1,3

Dengan cara yang sama seperti diatas, akan diperoleh tabel nilai probabilitas untuk s=2 pada Tabel 4.6. yaitu :

Tabel 4.6. Nilai probabilitas pilih kota s=2

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	5	0,113	0,134	0,000	0,328	0,000	0,090	0,180	0,062	0,054	0,039
2	1	0,000	0,275	0,137	0,179	0,129	0,075	0,125	0,000	0,045	0,036
3	4	0,155	0,155	0,324	0,000	0,000	0,074	0,137	0,066	0,051	0,038
4	3	0,000	0,000	0,142	0,210	0,179	0,121	0,254	0,000	0,052	0,041
5	3	0,000	0,247	0,000	0,161	0,116	0,067	0,112	0,100	0,040	0,033

Berdasarkan nilai probabilitas pilih kota tabel 4.6, dibangkitkan bilangan acak dan diperoleh *tabu list* saat s=2 pada Tabel 4.7 :

Tabel 4.7. Pengisian tabulist saat s=2

Semut	Bilangan acak	Pilih kota	Tabu
1	0,204	2	3,5,2
2	0,251	2	8,1,2
3	0,216	2	5,4,2
4	0,705	7	4,3,7
5	0,328	4	1,3,4

Selanjutnya dihitung nilai probabilitas untuk s=3 seperti pada Tabel 4.8 yaitu :

Tabel 4.8. Nilai probabilitas pilih kota s=3

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	2	0,280	0,000	0,000	0,182	0,000	0,105	0,221	0,131	0,045	0,036
2	2	0,000	0,000	0,142	0,210	0,179	0,121	0,254	0,000	0,052	0,041
3	2	0,297	0,000	0,131	0,000	0,000	0,111	0,235	0,139	0,048	0,038
4	7	0,138	0,240	0,000	0,000	0,228	0,198	0,000	0,101	0,054	0,041
5	4	0,000	0,183	0,000	0,000	0,383	0,088	0,162	0,078	0,060	0,045

Berdasarkan nilai probabilitas pada tabel 4.8 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.9 :

Tabel 4.9. Pengisian tabulist saat s=3

Semut	Bilangan acak	Pilih kota	Tabu
1	0,379	4	3,5,2,4
2	0,642	6	8,1,2,6
3	0,249	1	5,4,2,1
4	0,123	1	4,3,7,1
5	0,758	7	1,3,4,7

Selanjutnya untuk s=4 diperoleh tabel nilai probabilitas pilih kota pada Tabel 4.10. yaitu :

Tabel 4.10. Nilai probabilitas pilih kota s=4

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	4	0,297	0,000	0,000	0,000	0,000	0,142	0,263	0,127	0,098	0,073
2	6	0,000	0,000	0,140	0,166	0,199	0,000	0,347	0,000	0,083	0,065
3	1	0,000	0,000	0,259	0,000	0,000	0,142	0,236	0,210	0,085	0,068
4	1	0,000	0,410	0,000	0,000	0,192	0,112	0,000	0,166	0,067	0,054
5	7	0,000	0,278	0,000	0,000	0,264	0,230	0,000	0,118	0,062	0,048

Berdasarkan nilai probabilitas pada tabel 4.10 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.11 :

Tabel 4.11. Pengisian tabulist saat s=4

Semut	Bilangan acak	Pilih kota	Tabu
1	0,915	9	3,5,2,4,9
2	0,079	3	8,1,2,6,3
3	0,396	6	5,4,2,1,6
4	0,564	5	4,3,7,1,5
5	0,103	2	1,3,4,7,2

Selanjutnya untuk s=5 diperoleh tabel nilai probabilitas pilih kota pada Tabel 4.12. yaitu :

Tabel 4.12. Nilai probabilitas pilih kota s=5

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	9	0,141	0,000	0,000	0,000	0,000	0,136	0,153	0,105	0,000	0,465
2	3	0,000	0,000	0,000	0,449	0,275	0,000	0,137	0,000	0,080	0,059
3	6	0,000	0,000	0,179	0,000	0,000	0,000	0,445	0,186	0,107	0,083
4	5	0,000	0,352	0,000	0,000	0,000	0,238	0,000	0,164	0,142	0,103
5	2	0,000	0,000	0,000	0,000	0,329	0,222	0,000	0,278	0,096	0,076

Berdasarkan nilai probabilitas pada tabel 4.12 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.13 :

Tabel 4.13. Pengisian tabulist saat s=5

Semut	Bilangan acak	Pilih kota	Tabu
1	0,375	7	3,5,2,4,9,7
2	0,932	9	8,1,2,6,3,9
3	0,182	7	5,4,2,1,6,7
4	0,995	10	4,3,7,1,5,10
5	0,464	6	1,3,4,7,2,6

Selanjutnya untuk s=6 diperoleh tabel nilai probabilitas pilih kota pada Tabel 4.14. yaitu :

Tabel 4.14. Nilai probabilitas pilih kota s=6

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	7	0,289	0,000	0,000	0,000	0,000	0,414	0,000	0,212	0,000	0,086
2	9	0,000	0,000	0,000	0,186	0,195	0,000	0,153	0,000	0,000	0,466
3	7	0,000	0,000	0,392	0,000	0,000	0,000	0,000	0,314	0,166	0,127
4	10	0,000	0,145	0,000	0,000	0,000	0,138	0,000	0,114	0,604	0,000
5	6	0,000	0,000	0,000	0,000	0,405	0,000	0,000	0,295	0,169	0,132

Berdasarkan nilai probabilitas pada tabel 4.14 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.15 :

Tabel 4.15. Pengisian tabulist saat s=6

Semut	Bilangan acak	Pilih kota	Tabu
1	0,255	1	3,5,2,4,9,7,1
2	0,968	10	8,1,2,6,3,9,10
3	0,254	3	5,4,2,1,6,7,3
4	0,978	9	4,3,7,1,5,10,9
5	0,756	9	1,3,4,7,2,6,9

Selanjutnya untuk s=7 diperoleh tabel nilai probabilitas pilih kota pada Tabel 4.16. yaitu :

Tabel 4.16. Nilai probabilitas pilih kota s=7

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	1	0,000	0,000	0,000	0,000	0,000	0,337	0,000	0,501	0,000	0,163
2	10	0,000	0,000	0,000	0,349	0,356	0,000	0,295	0,000	0,000	0,000
3	3	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,354	0,371	0,274
4	9	0,000	0,368	0,000	0,000	0,000	0,356	0,000	0,276	0,000	0,000
5	9	0,000	0,000	0,000	0,000	0,254	0,000	0,000	0,137	0,000	0,608

Berdasarkan nilai probabilitas pada tabel 4.16 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.17 :

Tabel 4.17. Pengisian tabulist saat s=7

Semut	Bilangan acak	Pilih kota	Tabu
1	0,812	8	3,5,2,4,9,7,1,8
2	0,933	7	8,1,2,6,3,9,10,7
3	0,553	9	5,4,2,1,6,7,3,9
4	0,666	6	4,3,7,1,5,10,9,6
5	0,671	10	1,3,4,7,2,6,9,10

Selanjutnya untuk s=8 diperoleh tabel nilai probabilitas pilih kota pada Tabel 4.18. yaitu :

Tabel 4.18. Nilai probabilitas pilih kota s=8

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	8	0,000	0,000	0,000	0,000	0,000	0,730	0,000	0,000	0,000	0,270
2	7	0,000	0,000	0,000	0,435	0,565	0,000	0,000	0,000	0,000	0,000
3	9	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,184	0,000	0,816
4	6	0,000	0,579	0,000	0,000	0,000	0,000	0,000	0,421	0,000	0,000
5	10	0,000	0,000	0,000	0,000	0,618	0,000	0,000	0,382	0,000	0,000

Berdasarkan nilai probabilitas pada tabel 4.18 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.19 :

Tabel 4.19. Pengisian tabulist saat s=8

Semut	Bilangan acak	Pilih kota	Tabu
1	0,525	6	3,5,2,4,9,7,1,8,6
2	0,323	4	8,1,2,6,3,9,10,7,4
3	0,484	10	5,4,2,1,6,7,3,9,10
4	0,953	8	4,3,7,1,5,10,9,6,8
5	0,928	8	1,3,4,7,2,6,9,10,8

Selanjutnya untuk s=9 diperoleh tabel nilai probabilitas pilih kota pada Tabel 4.20. yaitu :

Tabel 4.20. Nilai probabilitas pilih kota s=9

Ant	Kota sekarang	Probabilitas pilih kota									
		1	2	3	4	5	6	7	8	9	10
1	6	0	0	0	0	0	0	0	0	0	1
2	4	0	0	0	0	1	0	0	0	0	0
3	10	0	0	0	0	0	0	0	1	0	0
4	8	0	1	0	0	0	0	0	0	0	0
5	8	0	0	0	0	1	0	0	0	0	0

Berdasarkan nilai probabilitas pada tabel 4.20 dibangkitkan bilangan acak dan diperoleh *tabu list* pada Tabel 4.21 :

Tabel 4.21. Pengisian tabulist saat s=9

Semut	Bilangan acak	Pilih kota	Tabu
1	0,577	10	3,5,2,4,9,7,1,8,6,10
2	0,209	5	8,1,2,6,3,9,10,7,4,5
3	0,723	8	5,4,2,1,6,7,3,9,10,8
4	0,567	2	4,3,7,1,5,10,9,6,8,2
5	0,401	5	1,3,4,7,2,6,9,10,8,5

Setelah semua kota tujuan dilewati, masing-masing semut akan kembali ke kota awal. Hasil rute yang dilewati oleh masing-masing semut ditunjukkan pada Tabel 4.22.

Tabel 4.22. Tour semut

Tour / Perjalanan
3,5,2,4,9,7,1,8,6,10,3
8,1,2,6,3,9,10,7,4,5,8
5,4,2,1,6,7,3,9,10,8,5
4,3,7,1,5,10,9,6,8,2,4
1,3,4,7,2,6,9,10,8,5,1

Berdasarkan Tabel 4.22. kemudian dihitung panjang jarak total *tour* masing-masing semut dengan persamaan **2.4** diperoleh hasilnya dan disajikan pada Tabel 4.23 :

Tabel 4.23. Jarak tour semut

Tour / Perjalanan	Total tempuh / Jarak total
3,5,2,4,9,7,1,8,6,10,3	555
8,1,2,6,3,9,10,7,4,5,8	445
5,4,2,1,6,7,3,9,10,8,5	460
4,3,7,1,5,10,9,6,8,2,4	438
1,3,4,7,2,6,9,10,8,5,1	489

Jarak total adalah jarak tempuh semut dari kota awal hingga kembali ke kota awal melewati setiap kota tepat satu kali.

4.3.4. Memperbarui Pheromone

Proses ini bertujuan untuk mengganti tabel pheromone awal dengan tabel pheromone yang baru. Tabel pheromone yang baru akan digunakan dalam iterasi berikutnya dengan persamaan :

$\tau_{ij}(t+1) = \rho \,.\, \tau_{ij}(t) \; + \; \sum_{k=1}^{m} \Delta\tau_{ij}{}^{k}$,

dengan $\Delta\tau_{ij}^{k} = Q/L_k$ jika (*i,j*) terletak dalam tour, dan $\Delta\tau_{ij}^{k} = 0$ jika (*i,j*) tidak terletak dalam *tour* dan L_k = panjang *tour*.

Untuk Q=100, $\rho = 0.5$, τ_{ij}= 0.1 didapatkan $\rho * \tau_{ij}$ = 0,05

Pasangan (1,2) terdapat pada *tour* ke-2 dan *tour* ke-3, mengakibatkan pheromone baru pasangan (1,2) = 0,05+($\frac{100}{445} + \frac{100}{460}$) = 0,492.

Pasangan (1,3) terdapat pada *tour* ke 5, mengakibatkan pheromone baru pasangan (1,3) = 0,05+($\frac{100}{489}$) = 0,2545.

Begitu juga untuk pasangan kota-kota yang lain dihitung dengan cara tersebut diatas dan hasil lengkap disajikan dalam Tabel 4.24 :

Tabel 4.24. Membuat tabel pheromone baru

	A	B	C	D	E	F	G	H	I	J
A	0,000	0,492	0,254	0,050	0,278	0,267	0,458	0,455	0,050	0,050
B	0,492	0,000	0,050	0,020	0,230	0,479	0,254	0,278	0,050	0,050
C	0,254	0,050	0,000	0,483	0,230	0,011	0,496	0,050	0,492	0,230
D	0,050	0,020	0,483	0,000	0,492	0,050	0,479	0,050	0,230	0,050
E	0,278	0,230	0,230	0,492	0,000	0,050	0,050	0,697	0,050	0,278
F	0,267	0,479	0,011	0,050	0,050	0,000	0,267	0,458	0,483	0,230
G	0,458	0,254	0,496	0,479	0,050	0,267	0,000	0,050	0,230	0,275
H	0,455	0,278	0,050	0,050	0,697	0,458	0,050	0,000	0,050	0,472
I	0,050	0,050	0,492	0,230	0,050	0,483	0,230	0,050	0,000	0,925
J	0,050	0,050	0,230	0,050	0,278	0,230	0,275	0,472	0,925	0,000

Nilai pada Tabel 4.24. menunjukkan bahwa nilai pheromone yang lebih besar menandakan lebih banyak semut yang memilih jalur tersebut. Melanjutkan proses ACO hingga *ite* terpenuhi dengan setiap *ite* akan membentuk tabel pheromone baru.

4.3.5. Seleksi

Setelah proses mendapatkan 5 *tabu list* pada hasil iterasi terakhir algoritma ACO, misalkan hasil akhir tersebut untuk *ite* = 1 maka didapatkan nilai total jaraknya yaitu $\sum_{k=1}^{5} f_k(x) = 555 + 445 + 460 + 438 + 489 = 2387$. Menghitung nilai *fitness* masing-masing kromosom dengan persamaan $\left(\sum_{k=1}^{5} f_k(x)\right) - f_k(x)$ dan didapatkan hasil seperti pada Tabel 4.25 :

Tabel 4.25. Menghitung nilai fitness

Kromosom	Proses	*fitness*
Semut1	$2387 - 555$	1832
Semut2	$2387 - 445$	1942
Semut3	$2387 - 460$	1927
Semut4	$2387 - 438$	1949
Semut5	$2387 - 489$	1898

Karena fungsi tujuan menemukan tour terpendek atau minimal, maka semakin pendek jarak total *tour* membuat nilai *fitness*nya semakin besar. Total nilai *fitness* adalah $F = \sum_{k=1}^{5} Semut_k = 9548$. Setelah menghitung total nilai *fitness*, langkah selanjutnya adalah menghitung *fitness* relatif (p_k) untuk setiap semut dengan rumus:

$$p_i = \frac{fitness_i}{F}, \quad i = 1,2,3,4,5$$

Setelah itu menghitung *fitness* kumulatif (q_i) untuk setiap semut dengan rumus:

$$q_i = \sum_{i=1}^{5} p_i\,, \quad i = 1,2,3,4,5$$

Hasil perhitungan *fitness* relatif dan *fitness* kumulatif disajikan pada Tabel 4.26 :

Tabel 4.26. Nilai fitness relatif dan kumulatif

Solusi	*Fitness* relatif (p_i)	*Fitness* kumulatif (q_i)
Semut1	0,191873	0,191873
Semut2	0,203393	0,395266
Semut3	0,201822	0,597088
Semut4	0,204127	0,801215
Semut5	0,198785	1

Untuk pemilihan solusi pada *roulette wheel* dengan cara membangkitkan nilai random r_i dengan *range* [0,1]. Pembangkitan nilai random r_i berfungsi dalam proses pemilihan calon induk untuk setiap semut.

4.3.6. Crossover

Diperoleh bilangan acak sebanyak 5 dan calon induk *crossover* dipilih dengan nilai bilangan acak mendekati jumlahan nilai kumulatifnya. Membangkitkan bilangan acak sebanyak 5 lagi dan mencari calon induk yang nilai bilangan acaknya kurang dari p_c. Hasil dari seleksi *crossover* bisa dilihat pada tabel 4.27 :

Tabel 4.27. Calon induk dan induk crossover

Bilangan Acak	q_k	*Calon Induk Crossover*	*Bilangan Acak*	*Induk*
0,649138	0,191873	Semut4	0,678	-
0,477192	0,395266	Semut3	0,236	Semut3
0,139184	0,597088	Semut1	0,595	-
0,689912	0,801215	Semut4	0,276	Semut4
0,282717	1	Semut2	0,346	Semut2

Semut2 dihapus dari induk baru hasil kawin silang sebab proses *crossover* membutuhkan 2 kromosom/semut. Solusi anak hasil *crossover* dengan metode *Single Point Crossover* dijelaskan pada Tabel 4.28.

Tabel 4.28. Solusi anak crossover

Semut	Induk										Kota Awal	Jarak
3	E	D	B	A	F	G	C	I	J	H	E	460
4	D	C	G	A	E	J	I	F	H	B	D	438
	Random Bilangan [1,*jml_kota*] = 7											
	Solusi Anak *Crossover*											
3	E	D	B	A	F	G	I	F	H	B	E	
4	D	C	G	A	E	J	C	I	J	H	D	
	Memperbarui Solusi Anak *Crossover*											
3	E	D	J	A	C	G	I	F	H	B	E	580
4	D	F	G	A	E	B	C	I	J	H	D	490

Pembaruan solusi anak *crossover* bertujuan agar dalam satu kromosom anak tidak terdapat lokus yang sama. Lokus yang sama berarti mengunjungi kota tidak tepat satu kali, bukan permasalahan DTSP.

4.3.7. Mutasi

Diperoleh bilangan acak sebanyak 5 dan calon induk mutasi dipilih dengan nilai bilangan acak mendekati jumlahan nilai kumulatifnya. Membangkitkan bilangan acak sebanyak 5 lagi dan mencari calon induk yang nilai bilangan acaknya kurang dari p_m. Hasil dari seleksi mutasi bisa dilihat pada tabel 4.29 :

Tabel 4.29. Calon induk dan induk mutasi

Bilangan Acak	q_k	*Calon Induk Mutasi*	*Bilangan Acak*	*Induk*
0,10562	0,191873	Semut1	0,76617	-
0,59951	0,395266	Semut4	0,52788	-
0,00576	0,597088	Semut1	0,00789	Semut1
0,48067	0,801215	Semut3	0,91957	-
0,36584	1	Semut2	0,22669	-

Solusi anak hasil mutasi dengan menggunakan metode *Resiprocal Exchange Mutation* dijelaskan pada Tabel 4.30.

Tabel 4.30. Solusi anak mutasi

Semut	Induk										Kota Awal	Jarak
1	C	E	B	D	I	G	A	H	F	J	C	555
	Random 2 Bilangan [1,jml_kota] = 2 dan 5											
1	C	E	B	D	I	G	A	H	F	J	C	555
	Solusi Anak Mutasi											
1	C	I	B	D	E	G	A	H	F	J	C	541

Tidak terdapat pembaruan solusi anak mutasi, sebab pada metode *Resiprocal Exchange Mutation* hanya pertukaran 2 lokus yang berbeda. Terjamin permasalahan DTSP tetap berjalan dengan tidak adanya lokus ganda.

4.3.8. Memperbarui Populasi

Setelah mendapatkan solusi anak dari proses *crossover* dan mutasi, solusi baru tersebut akan di*merge* dengan solusi sebelumnya. Dan diseleksi sebanyak 5 saja untuk diproses pada iterasi GA berikutnya dengan diurutkan berdasarkan jarak paling pendeknya dan didapatkan hasil pada Tabel 4.31 :

Tabel 4.31. Membentuk populasi baru

Semut	*Jarak*	*Urutkan Ascending*	$Semut^{new}$
1	555	438	4
2	445	445	2
3	460	460	3
4	438	489	5
5	489	490	7
6	580	541	
7	490	555	
8	541	580	

$Semut^{new}$ adalah populasi baru yang akan memasuki proses GA iterasi berikutnya. $Semut^{new}$ akan diperbarui sampai *ite* yang diinginkan terpenuhi.

4.3.9. Menambah atau Mengurangi Kota Tujuan

Misalkan *ite* yang diinginkan sudah terpenuhi dan diperoleh *tour* terpendeknya untuk satu semut adalah sebagai berikut :

$$4 \rightarrow 3 \rightarrow 7 \rightarrow 1 \rightarrow 5 \rightarrow 10 \rightarrow 9 \rightarrow 6 \rightarrow 8 \rightarrow 2 \rightarrow 4$$

dengan jarak totalnya adalah 438.

1) Mengurangi kota tujuan

Diketahui rute yang dilalui semut adalah sebagai berikut :

Tour Awal										Kota Akhir
D	C	G	A	E	J	I	F	H	B	D

Misalkan kota yang akan dihapus dari kota tujuan adalah kota **J**. Maka kunjungan semut ke kota **J** setelah kota **E** dibatalkan dan semut membentuk pheromone baru dengan jumlah kota yaitu kota **E**, kota **I**, kota **F**, kota **H** dan kota **B** sebanyak 5 kota.

Tour Akhir									Kota Akhir
D	C	G	A	E	I	F	H	B	D

Dibentuk tabel pheromone awal seperti pada Tabel 4.32 :

Tabel 4.32. Pheromone awal hapus kota

	E	I	F	H	B
E	0	0,2	0,2	0,2	0,2
I	0,2	0	0,2	0,2	0,2
F	0,2	0,2	0	0,2	0,2
H	0,2	0,2	0,2	0	0,2
B	0,2	0,2	0,2	0,2	0

Tabel pheromone awal ini akan diproses dengan algoritma ACO dan algoritma genetika dengan kota awal adalah kota **E** hingga *ite* terpenuhi.

2) Menambah kota tujuan

Diketahui rute yang dilalui semut adalah sebagai berikut :

Tour Awal										Kota Akhir
D	C	G	A	E	J	I	F	H	B	D

Misalkan terdapat penambahan kota tujuan yaitu kota **K** saat semut telah mencapai kota **I**. Maka semut takkan langsung menuju kota **F** setelah kota **I**, semut akan membentuk *pheromone* baru dengan jumlah kota 5, yaitu kota **I**, kota **K**, kota **F**, kota **H** dan kota **B**.

Tour Awal											Kota Akhir
D	C	G	A	E	J	I	K	F	H	B	D

Dibentuk tabel pheromone awal seperti pada Tabel 4.33 :

Tabel 4.33. Pheromone awal tambah kota

	I	K	F	H	B
I	0	0,2	0,2	0,2	0,2
K	0,2	0	0,2	0,2	0,2
F	0,2	0,2	0	0,2	0,2
H	0,2	0,2	0,2	0	0,2
B	0,2	0,2	0,2	0,2	0

Tabel pheromone awal ini akan diproses dengan algoritma ACO dan algoritma genetika dengan kota awal adalah kota **I** hingga *ite* terpenuhi.

Sebagai contoh pengerjaan *update* kota dalam prosedur mengurangi (*delete*) kota tujuan adalah sebagai berikut :

i. Berdasarkan tabel 4.32, kota awal adalah kota **E**.

ii. Menghitung nilai probabilitas pilih kota sebanyak $(n-1) = (5-1) = 4$ langkah atau s = 4. Hasil perhitungan dari proses ACO disajikan pada tabel 4.34.

Tabel 4.34. Proses ACO pencarian tour update hapus kota

Semut	Kota awal	Kota ke-	Nilai probabilitas pilih kota saat s=1					Random	Tour
			E	I	F	H	B		
1	E	1	0	0,158	0,265	0,183	0,393	0,67	E,B
2	E	1	0	0,158	0,265	0,183	0,393	0,155	E,I
3	E	1	0	0,158	0,265	0,183	0,393	0,715	E,B
4	E	1	0	0,158	0,265	0,183	0,393	0,365	E,F
5	E	1	0	0,158	0,265	0,183	0,393	0,547	E,H
Semut	Kota sekarang	Kota ke-	Nilai probabilitas pilih kota saat s=2					Random	Tour
			E	I	F	H	B		
1	B	5	0	0,16	0,373	0,467	0	0,531	E,B,F
2	I	2	0	0	0,356	0,276	0,368	0,863	E,I,B
3	B	5	0	0,16	0,373	0,467	0	0,371	E,B,F
4	F	3	0	0,194	0	0,339	0,467	0,289	E,F,H
5	H	4	0	0,14	0,316	0	0,544	0,552	E,H,B
Semut	Kota sekarang	Kota ke-	Nilai probabilitas pilih kota saat s=3					Random	Tour
			E	I	F	H	B		
1	F	3	0	0,464	0	0,536	0	0,883	E,B,F,H
2	B	5	0	0	0,444	0,556	0	0,645	E,I,B,H
3	F	3	0	0,464	0	0,536	0	0,203	E,B,F,I
4	H	4	0	0,205	0	0	0,795	0,241	E,F,H,B
5	B	5	0	0,301	0,699	0	0	0,427	E,H,B,F
Semut	Kota sekarang	Kota ke-	Nilai probabilitas pilih kota saat s=4					Random	Tour
			E	I	F	H	B		
1	H	4	0	1	0	0	0	0,901	E,B,F,H,I
2	H	4	0	0	1	0	0	0,687	E,I,B,H,F
3	I	2	0	0	0	1	0	0,321	E,B,F,I,H
4	B	5	0	1	0	0	0	0,445	E,F,H,B,I
5	F	3	0	1	0	0	0	0,421	E,H,B,F,I

iii. Prosedur seleksi *roulette wheel*, *crossover* dan mutasi

iv. Menghitung jarak total tournya, nilai *fitness* relatif dan kumulatif tiap individu semut. Hasil perhitungan disajikan pada tabel 4.35.

Tabel 4.35. Evaluasi, nilai fitness relatif dan kumulatif

Semut	Jarak tour	Evaluasi	Fitness relatif	Fitness kumulatif
1	316	1222	0,198635	0,198635
2	295	1243	0,202048	0,400683
3	341	1197	0,194571	0,595254
4	290	1248	0,202861	0,798114
5	296	1242	0,201886	1
Total Jarak = 1538				
Total evaluasi = 6152				

Setelah menghitung nilai *fitness* relatif dan kumulatif akan dicari anak-anak yang dibentuk oleh induk terpilih dari proses *crossover* dan mutasi. Pencarian induk terpilih dapat dilihat pada tabel 4.36.

Tabel 4.36. Calon induk dan induk crossover saat update

Bilangan Acak	q_k	*Calon Induk Crossover*	*Bilangan Acak*	*Induk*
0,602	0,1986	Semut4	0,314	Semut4
0,608	0,4006	Semut4	0,0385	Semut4
0,569	0,5955	Semut3	0,255	Semut3
0,268	0,7981	Semut2	0,692	-
0,686	1	Semut4	0,533	-

Semut4 dan Semut4 terpilih untuk memasuki prosedur *crossover*. Kemudian akan didapatkan solusi anak hasil *crossover* dengan prosedur pada tabel 4.37.

Tabel 4.37. Manual solusi anak crossover

Semut	Induk										Kota Akhir	Jarak
4	E		F		H		B		I		D	290
4	E		F		H		B		I		D	290
	Random Bilangan [1,*jml_kota*] = 2											
	Solusi Anak *Crossover*											
4	E		F		H		B		I		D	
4	E		F		H		B		I		D	
	Memperbarui Solusi Anak *Crossover*											
4	E		F		H		B		I		D	290
4	E		F		H		B		I		D	290

Dan bersamaan dengan proses *crossover* terdapat pula prosedur mutasi yang disajikan pada tabel 4.38.

Tabel 4.38. Calon induk dan induk mutasi saat update

Bilangan Acak	q_k	*Calon Induk Mutasi*	*Bilangan Acak*	*Induk*
0,499	0,1986	Semut3	0,731	-
0,235	0,4006	Semut2	0,542	-
0,754	0,5955	Semut4	0,649	-
0,222	0,7981	Semut2	0,709	-
0,837	1	Semut5	0,574	-

Berdasarkan tabel 4.38. tidak terdapat induk semut yang terpilih untuk memasuki prosedur mutasi.

v. Setelah prosedur *crossover* dan mutasi selesai, anak dan induk akan digabung jadi satu serta diurutkan berdasarkan solusi paling baik (*ascending*) dan diambil sebanyak jumlah semut awal untuk iterasi berikutnya. Prosedur mendapatkan populasi baru semut dapat dilihat pada tabel 4.39.

Tabel 4.39. Membentuk populasi baru saat update

$Semut$	$Tour$	$Jarak\ total$	$Urutkan\ Ascending$	$Semut^{new}$
1	E,B,F,H,I,D	316	290	4
2	E,I,B,H,F,D	295	290	6
3	E,B,F,I,H,D	341	290	7
4	E,F,H,B,I,D	290	295	2
5	E,H,B,F,I,D	296	296	5
6	E,F,H,B,I,D	290	316	
7	E,F,H,B,I,D	290	341	

vi. Hasil akhir update jarak total dengan mengurangi kota tujuan adalah :

$$= d_{D,C} + d_{C,G} + d_{G,A} + d_{A,E} + (hasil\ update\ kurangi\ kota\ )$$

$$= d_{D,C} + d_{C,G} + d_{G,A} + d_{A,E} + d_{E,B} + d_{B,F} + d_{F,H} + d_{H,I} + d_{I,D}$$

$$= 11 + 36 + 33 + 32 + 27 + 40 + 55 + 124 + 70$$

$$= 428.$$

Jadi jarak DTSP untuk kurangi kota **J** untuk satu kali iterasi adalah 428.

vii. Prosedur pengerjaan *update* tambah kota sama dengan *update* kurangi kota, namun dengan tabel pheromone awal yang berbeda.

4.4. Implementasi Program pada Contoh Kasus

Program dari *hybrid* algoritma ACO dan algoritma genetika yang telah dibuat digunakan untuk menyelesaikan *Dynamic Travelling Salesman Problem* (DTSP) dengan menggunakan *source code* pada **Lampiran 4**. Data yang digunakan terdapat pada **Lampiran 3**. Untuk menyelesaikan permasalahan *Dynamic Travelling Salesman Problem* (DTSP) menggunakan data 50 kota tujuan awal oleh *salesman* dan berubah dengan adanya *update* penambahan 5 kota tujuan lagi saat *salesman* berada dalam, oleh karena itu penulisan kota tujuan adalah 50+5. *Update* kota dengan penambahan atau pengurangan kota tujuan tidak boleh dari kota awal, sebab kemungkinan solusi berubah seluruhnya.

Menggunakan nilai parameter konstan $\rho = 0.7, Q = 1000$. Hasil yang bervariasi dengan perbedaan nilai parameter $\alpha, \beta, p_c, p_m,$ banyak iterasi dan jumlah *pop_size* disajikan pada Tabel 4.40.

Tabel 4.40. Perbandingan parameter dengan 50+5 kota

Banyak Kota	Iterasi	Pop_size	α	β	p_c	p_m		
						0.01	0.1	0.5
50+5	100	10	0.5	5	0.6	20516	20745	20273
					0.7	20611	20575	20396
					0.9	20424	20400	20368
			1	1	0.6	21937	21915	21733
					0.7	22951	21063	21044
					0.9	22304	22335	22084
			5	0.5	0.6	44426	44064	42780
					0.7	44586	43345	43182
					0.9	43918	43427	42554
		30	0.5	5	0.6	20278	20121	20035
					0.7	20878	20676	20653
					0.9	19846	20112	20019
			1	1	0.6	20510	20961	20104
					0.7	22087	21470	21362
					0.9	20358	20474	20569
			5	0.5	0.6	32590	34152	36705
					0.7	34157	33664	34249
					0.9	34155	34270	34105
	500	10	0.5	5	0.6	20345	20369	20248
					0.7	20150	20790	20212
					0.9	20420	20409	20189
			1	1	0.6	20951	20914	20465
					0.7	22712	21495	21307
					0.9	20993	20576	20278
			5	0.5	0.6	46913	38849	37132
					0.7	41227	43769	38347
					0.9	42558	52620	41314
		30	0.5	5	0.6	20267	20207	19674
					0.7	20750	20163	20137
					0.9	20680	20050	19830
			1	1	0.6	20687	20437	19898
					0.7	20372	21057	20543
					0.9	20507	20266	20463
			5	0.5	0.6	32758	33095	32736
					0.7	33127	33905	32053
					0.9	33278	33288	32187

Berdasarkan Tabel 4.40. dapat diketahui parameter yang akan membuat solusi makin kecil atau perjalanan paling pendek adalah parameter nilai $\alpha = 0.5$, nilai $\beta = 5$, banyak iterasi = 500, *pop_size* = 30

serta $p_c = 0.6$ dan $p_m = 0.5$ dengan jarak totalnya 19674. Dan hasil dari *generate* rute terbaik TSP sebanyak 50 kota adalah :

18 → 3 25 21 15 17 23 37 35
20 9 5 48 0 46 31 10 16
14 44 22 34 26 11 19 6 8
33 45 28 2 42 13 47 40 29
38 4 36 32 12 1 43 49 39
24 27 7 41 30 → 18

dengan jarak tempuh minimal = 17563.0

Update kota sebanyak 5 kota yang ditambahkan saat mencapai kota ke-25 adalah :

50 51 52 53 54

Sehingga kombinasi kota yang *update* :

19 *50 51 52 53 54* 6 8 33
45 28 2 42 13 47 40 29 38
4 36 32 12 1 43 49 39 24
27 7 41 30

dengan kota awal adalah kota ke-19.

Hasil terbaik yang didapatkan adalah :

18 →	3	25	21	15	17	23	37	35
20	9	5	48	0	46	31	10	16
14	44	22	34	26	11	19	8	6
54	33	28	45	2	42	40	47	13
29	38	*51*	4	36	12	32	49	43
1	*53*	39	24	*50*	27	7	41	30
52 →	18							

dengan total jarak tempuh minimal ter*update* = 19674.0

Jadi, dapat disimpulkan bahwa parameter yang membuat solusi jarak semakin pendek adalah dengan membuat nilai $\alpha < \beta$, memperbanyak jumlah koloni semut (*pop_size*) sehingga kemungkinan rute yang terbentuk semakin bervariasi, dan banyaknya iterasi sebab semakin banyak iterasi solusi yang didapatkan semakin baik. Perbedaan nilai laju *crossover* dan laju mutasi cenderung kurang berpengaruh pada jarak total rute.