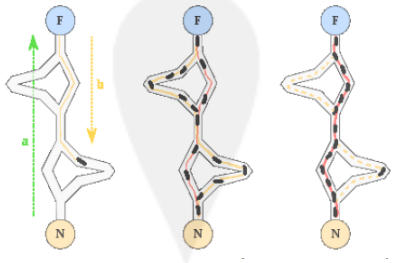
BAB III
LANDASAN TEORI

3.1 Algoritma ACO

Algoritma ACO telah di perkenalkan oleh Macro Dorigo di awal tahun 1990an. Macro terinspirasi oleh perlakuan semut dalam pencarian makanan. Bagaimana semut dapat menemukan jalur terpendek antara sumber makanan dengan sarang semut. Saat mencari makanan, semut awalnya menjelajahi daerah sekitarnya dengan cara acak (Alhanjouri & Alfaran, 2012)

Secara jelas cara kerja semut menemukan rute terpendek dalam ACO adalah sebgai berikut : secara alamiah semut mampu menemukan rute terpendek dalam perjalanan dari sarang ke tempat sumber makanan. Koloni semut dapat menemukan rute berdasarkan jejak kaki pada lintasan yang telah di lalui. Semakin banyak semut yang melalui suatu lintasan, maka akan semakin jelas jejak kakinya. Proses peninggalan jejak kaki yang dilalui semut di namakan *(pheromone)*. Jika lintasan yang dilalui semut dalam jumlah sedikit maka semakin lama akan semakin berkurang kepadatan semut. Sebaliknya jika lintasan yang dilalui semut dalam jumlah banyak maka semakin lama semakin bertambah kepadatan semut yang melewatinya atau bahkan semua semut akan melalui lintasan tersebut (Alhanjouri & Alfaran, 2012).

Setiap semut pada semua titik *graph* (dalam hal ini titik-titik yang akan dikunjungi) akan bergerak mengunjungi seluruh titik. Setiap semut akan membuat jalur masing-masing sampai kembali tempat di mana mereka di tempatkan (Leksono, 2009). Berikut ini merupakan gambar konsep ACO yang mencontoh semut dalam menemukan rute terpendek.



Gambar 3.1. Konsep ACO
Sumber : Alhanjouri & Alfarra (2012)

Semut pada mulanya mengambil posisi masing-masing kemudian bergerak menuju lokasi tujuan tingkat *pheromone* yang paling tinggi itulah yang akan di pilih semut untuk perjalanan selanjutnya sedangkan *pheromone* yang rendah perlahan akan ditinggalkan oleh semut sehingga jejak tersebut akan menghilang.

Menurut Dorigo dalam penemuannya tahun 1996 tentang algoritma semut maka secara informal, ACO bekerja sebagai berikut : setiap semut memulai tournya melalui titik yang dipilih secara acak (setiap semut memiliki titik awal yang berbeda). Secara berulang kali, satu persatu titik yang di kunjungi oleh semut dengan tujuan untuk menghasilkan sebuah tour. Pemilihhan titik-titik yang akan di laluinya didasarkan pada suatu fungsi probailitas, dinamai aturan trnasisi status (*state transition rule*), dengan mempertimbangkan *visibility* (invers dari jarak) titik tersebut dan jumlah *pheromone* yang terdapat pada ruas yang menghubungkan titik tersebut. Semut lebih suka bergerak menuju ke titik-titik yang dihubungkan dengan ruas yang pendek dan memilih tingkat *pheromone* yang tinggi.

Menurut Leksono (2009) dalam skripsinya yang berjudul *"algoritma ant colnoy optimization untuk menyelesaikan travelling salesman problem (TSP)"* setiap semut memiliki sebuah memori, dinamai *tabulist* yang berisi titik yang telah di kunjungi pada setiap tour. *Tabulist* ini mencegah semut untuk mengunjungi titik-titik sebelumnya yang telah di kunjungi selama tour tersebut berlangsung, yang membuat solusinya mendekati optimal. Setelah semua semut menyelesaikan tour mereka dan *tabulist* mereka menjadi penuh, sebuah aturan *pheromone* diterapkan pada setiap semut. Penguapan *pheromone* pada semua ruas dilakukan, kemudia setiap semut menghitung panjang tour yang telah dilakukan lalu meninggalkan sejumlah *pheromone* pada edge-edge yang merupakan bagian dari tour mereka yang sebanding dengan kualitas dari solusi yang mereka hasilkan.

Semakin pendek sebuah tour yang dihasilkan oleh setiap semut, jumlah *pheromone* yang ditinggalkan pada edge-edge yang dilaluinya pun semakin besar.

Dengan kata lain, edge yang merupakan bagian dari tour-tour terpendek adalah edge-edge yang menerima jumlah *pheromone* yang lebih besar. Hal ini menyebabkan edge-edge yang diberi *pheromone* lebih banyak akan lebih diminati pada tour-tour selanjutnya, dan sebaliknya edge-edge yang tidak di beri *pheromone* menjadi kurang diminati. Dan juga, rute terpendek yang ditemukan oleh semut disimpan dan semua *tabu list* yang ada dikosongkan kembali.

Peranan utama dari penguapan *pheromone* adalah untuk mencegah stagnasi, yaitu situasi dimana semua semut berakhir dengan melakukan tour yang sama. Proses diatas kemudia diulangi sampai tour-tour yag dilakukan mencapai jumlah maksimum atau system ini menghasilakn perilaku stagnasi di mana system ini berhenti untuk mencari solusi alternatif.

Nilai parameter mempengaruhi nilai di mana tau probabilitas dari kota i ke kota j. semakin besar nilai keduanya semakin besar pula probabilitas dari kota yang sekarang ke kota berikutnya. Ini berarti nilai parameter α dan β berbanding lurus dengan nilai. Nilai parameter ρ akan mempengaruhi nilai t merupakan intensitas *pheromone*. Intensitas *pheremone* setiap kota berbeda-beda. Setiap iterasi yang di lakukan menyebabkan perubahan pada intensitas tersebut (Lukas & Ariwibowo, 2007).

Menurut beberapa penelitian penelitian ,Jufri & Santoso (2014), Lestari & Sari (2013), Mutakhiroh (2007), Sanjaya (2014), (Leksono, 2009) ACO memerlukan beberapa variabel sebelum melakukan pencarian jalur terpendek yaitu :

Tahap I

Inisialisasi harga parameter.

a) (τ_{ij}) adalah intensitas jejak *pheromone* semut
b) (n) adalah banyak kota termasuk koordinat (x,y) atau jarak antar kota (d_{ij})

Rumus untuk menentukan jarak adalah

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (3.1)$$

c) Q adalah tetapan siklus semut.

d) α adalah tetapan pengendali intesitas jejak semut, nilai $\alpha > 0$

e) β adalah tetapan pengendali visibilitas, nilai $\beta > 0$

f) Visibilitas antar kota $(\eta_{ij}) = 1/d_{ij}$

g) m adalah jumlah semut.

h) ρ adalah tetapan penguapan jejak semut, dimana $0 \leq \rho \leq 1$

i) NCmax adalah jumlah siklus maksimum.

Tahap 2

Pengisian *node* pertama ke dalam *tabu list*.

Tabu list digunakan untuk menyimpan daftar urutan *node-node* yang sudah di kunjungi setiap semut. Setiap kali semut berkunjung ke suatu kota maka elemen *tabu list* akan bertambah satu, seterusnya sampai *tabu list* penuh.

Tahap 3

Memberikan nilai bobot untuk setiap parameter yang digunakan yaitu (nilai τ_{ij} , alfa, beta,rho,q, dan NcMax)

Tahap 4

Hitung persamaan probabilitas untuk menentukan kota tujuan

$$P_{ij}^{k} = \begin{cases} \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{u \epsilon j_r^k}[\tau_{ik}]^{\alpha}[\eta_i]^{\beta}} \end{cases} \quad (3.2)$$

Dimana :

τ_{ij} : jumlah feromon pada sisi dari simpul i ke simpul j

P_{ij}^{k} : probabilitas semut k untuk berpindah dari simpul i ke simpul j

η_{ij} : visibilitas antar kota i ke simpul j menggunakan rumus $\eta_{ij}) = 1/d_{ij}$, dimana $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ (jika hanya di ketahui koordinat titiknya saja)

j_r^k : himpunan titik yang akan dikunjungi oleh semut k yang sedang berada pada titik r (untuk membuat soluisnya mendekati optimal)

u : simpul yang berada dalam J_r

Tahap 5

Apabila telah memilih node yang dituju, node tersebut disimpan ke dalam bentuk tabu list untuk menyatakan bahwa node tersebut telah emnajdi bagian dari membangun solusi. Setelah itu intensitas *pheromone* tersebut diubah dengan menggunakan (3.3) dan (3.4) maka perubahan tersebut di namakan perubahan *pheromone* local. Aturan transisi kembali dilakukan sampai node tujuan tercapai. Aturan pembaruan *pheromone* diimplementasikan ke dalam rumus

$$\tau(t,v) = (1-\rho).\tau(t,v) + \rho.\Delta\tau(t,v) \quad (3.3)$$

$$\Delta\tau(t,v) = \frac{1}{Lnn^c} \quad (3.4)$$

Dimana

L_{nn} = panjang tur yang dperoleh

C = jumlah lokasi

$\Delta\tau$ = perubahan *pheromone*

ρ = parameter dengan nilai > 0

Tahap 6

Pembaharuan *pheromone* pada node-node yang termuat dalam path terbaik menggunakan persamaan (3.5) dan (3.6) disebut pembaharuan *pheromone* global

$$\tau(t,v) = (1-\alpha).\tau(t,v) + \alpha.\Delta\tau(t,v) \quad (3.5)$$

$$\Delta\tau(t,v) = \begin{cases} L_{gb}^{-1} \text{ jika (t,v)} \in \text{tour \_terbaik} \\ 0 \end{cases} \quad (3.6)$$

Dimana

$\tau(t,v)$ = Nilai *pheromone* setelah mengalami pembaruan local

L_{gb} = Panjang tur terpendek pada akhir siklus

$\Delta\tau$ = perubahan *pheromone*

α = parameter dengan nilai >0

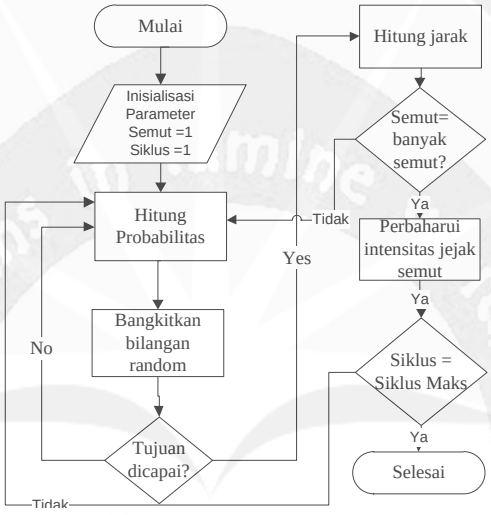
Setelah itu pengosongan tabu list, tabu list dikosongkan untuk diisi lagi dengan urutan node yang baru. Algoritma diulang dari langkah dua dengan harga parameter intensitas *pheromone* yang sudah di baharui. Setelah semua proses telah di uji maka (jumlah siklus maksimum sudah terpenuhi) maka akan di dapatkan path terbaik.

Berikut ini merupakan *pseducode* tentang algoritma ACO yaitu :

```
Deklarasi
alfa, beta, rho ;
m, Q,;
Deskripsi
Set parameter (alfa,beta, rho,m,Q)
while stopping criterion not satisfied do
position each ant in a strating node
repeat
      for each ant do
choose next node by appling the state
transition rule
apply local pheromone update
end for
```

```
      until every ant has built a solution
update best solution
apply  global pheromone update
end while
Output (Jarak terpendek)
```

Sesuai penjelasan tahap *pseudocode* diatas maka dapat di jelaskan melalui diagram alir yaitu :



Gambar 3.2. Diagram alir ACO
Sumber : Nugraha (2012)

Adapun langkah-langkah dalam menyelesaikan ACO menurut Leksono, (2009), Pitralisa & Wirayuda, (2011) adalah sebagai berikut :

Langkah I adalah Penetapan parameter dan nilai *pheromone* awal

a) Insialisasi harga parameter yang digunakan
    1) Tentukan banyaknya titik (n) atau jumlah kota
    2) Tetapkan nilai intensitas jejak semua antar titik dan perubahannya (τ_{ij}) nilai ini akan terus berubah pada mulai dari iterasi pertama sampai iterasi maksimum yang ditentukan

3) Tentukan nilai pengendali intensitas jejak semut (α) dan pengendali visibilitas(β) dimana nilai α,β > 0
4) Tentukan nilai penguapan *pheromone* (ρ), nilainya $0 \leq \rho \leq 1$ hal ini untuk mencegah jumlah *pheromone* yang tak terhingga
5) Tentukan jumlah semut
6) Tentukan jumlah iterasi maksimum (NcMax)
7) Hitung visibilitas antar titik (η_{ij}) $= 1/d_{ij}$)

b) Setelah di lakukan proses inisialisasi maka dilakukan proses penempatan semut pada titik pertama secara acara menurut jumlah n kota. kemudian setiap semut memulai perjalanan dari titik asal ke titik tujuannya. Setelah itu dari titik kedua semut melakukan perjalan ke titik selanjutnya. Perjalanan semut berlangsung terus menerus sampai semua titik dikunjungi satu persatu.

c) Selagi melakukan perjalan untuk mencari solusi pencarian rute terpendek semut mengunjungi sisi-sisi dan mengubah tingkat *pheromone* pada sisi-sisi tersebut dengan menerapkan aturan *pheromone local* . Tujuan dari update *pheromone* adalah meningkatkan nilai *pheromone* yang berkaitan dengan lintasan yang menjanjikan atau bagus. Menurut Siswanto, 2013 dalam penelitian yang berjudul "pengembangan model pemilihan rute jaalan raya berdasarkan perilaku pengguna menggunakan ACO" dikemukakan bahwa setelah semut-semut menyelesaikan satu rute mereka makan semut akan menambahkan *pheromone* pada ruas-ruas yang sudah dikunjungi, yaitu ruas-ruas yang masuk dalam rutenya. Sementara ruas yang lain tidak berubah tingkat *pheromonenya.*. *Jumlah pheromone* yang ditambahkan $\Delta\tau(t, v)$dalam setiap ruas yang di kunjungi (t, v) oleh semut terbaik berbanding terbalik dengan jarak total rute; semakin pendek rute semakin tinggi jumlah *pheromone* yang ditambahkan pada ruas-ruasnya. Cari ini memang di maksudkan untuk meniru bagaimana *pheromone* ini

diakumulasi di dalam kejadian nyata dimana dipengaruhi oleh panjang lintasan dan kontinius waktu semut menjalani lintasan. Pengaruh dari pembaruan lokal ini adalah untuk membuat tingkat kemenarikan ruas-ruas yang ada berubah secara dinamis: setiap kalai seekor semut menggunkan sebuah ruas maka ruas ini dengan segera akan segera berkurang tingkat ketertarikannya (karena pada ruas tersebut jumlah *pheromone* berkurang akibat penguapan). Ini secara tidak langsung akan membuat semut yang akan memilih ruas-ruas lain yang belum dikunjungi. Sehingga semut-semut tidak akan memiliki kecenderungan membuat rute yang berbeda makan akan terdapat kemungkinan yang lebih tinggi untuk mendapatkan salah satu rute yang lebih baik daripada jika mereka menempuh rute yang sama. Aturan *pheromone* lokal ini berguna untuk mencegah ruas-ruas yang bagus tidak dipilih oleh semua semut: setiap kali suatu ruas dipilih oleh seekor semut jumlah *pheromone* berubah dengan menerapkan aturan pembaruan lokal.

d) Setelah melakukan *pheromone* secara local maka di lakukan perhitungan *pheromone* secara global. Perhitungan ini hanya dilakukan oleh semut yang membuat tur terpendek sejak permulaan percobaan. Pada akhir sebuah iterasi, setelah semua semua menyelesaikan tur mereka, sejumlah *pheromone* ditaruh pada ruas-ruas yang dilewati oleh seekor semut yang telah menemukan tur terbaik.
e) *Tabu list dikosongkan* kembali untuk diisi dengan urutan titik baru pada iterasi selanjutnya, jika NcMax belum tercapai atau belum terjadi konvergensi ()

3.1.1 Pengaruh nilai alfa terhadap jarak

Menurut penelitian Pitralisa & Wirayuda, (2011) dengan *judul "implementasi algoritma ant colony dengan multi agent pada kasus pencarian jalur terpendek"* dikemukakan bahwa nilai alfa pada umumnya di gunakan untuk mengontrol

intensitas *pheromone* (τ_{ij}). Menurut penelitian terbaru yang dilakukan oleh Lutfiani Semakin besar nilai alfa maka kemungkinan pengaruh *pheromone* akan semakin kuat.

3.1.2 Pengaruh nilai beta terhadap jarak

Beta digunakan untuk mengontrol visibilitas. Visibilitas merupakan invers dari jarak. Tujuan dari adanya visibilitas juga terbukti yakni mempengaruhi sensitivitas dari pembagian *pheromone* antara simpul. Nilai (τ_{ij}) Yang kecil dan alfa yang hanya digunakan pada proses eksplorasi juga menjadi penyebab nilai visibilitas lebih berperan dalam perhitungan untuk menghasilkan jarak terpendek (Pitralisa & Wirayuda, 2011). Semakin besar nilai beta maka kemungkinan pengaruh pemilihan jarak selanjutnya akan semakin kuat .

3.1.3 Pengaruh nilai rho terhadap jarak

Rho merupakan koefisien penguapan/evaporasi *pheromone* dan nilai rho digunakan untuk menentukan seberapa besar feromon yang akan hilang dan yang bertambah jika suatu jalur tidak dilewati atau dilewati. Parameter rho digunakan untuk menghindari akumulasi terbatas dari *pheromone trails* (Dorigo, 1996)

3.2 Algoritma Genetika

Menurut Asim,dkk. (2014), AG pertama kali diperkenalkan oleh John Holland dari Universitas Michigan (1975) merupakan suatu algoritma pencarian heuristik yang di terapkan pada banyak bidang seperi bidang-bidang fisika, kimia, bioinformatika, matematika, ekonomi, ilmu komputer serta filogenetika dan sekarang telah digunakan secara luas untuk masalah optimasi.

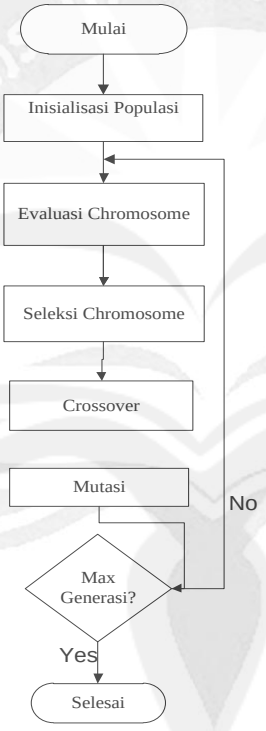
Algoritma genetika dikembangkan pada tahun 1960 oleh John Holland dengan mengambil prinsip Darwin yaitu mengubah populasi dari individu menjadi nilai *fitness* untuk menjadi generasi baru kemudian dilakukan *crossover* dan mutasi dan algoritma genetic berhasil menjawab berbagai masalah optimasi meskipun masih

ada beberapa factor yang membatasi keberhasilan algoritma genetika (Rexhepi ,dkk.(2013) dan Khan,dkk. (2009).

Selanjutnya menurut penelitian yang di lakukan oleh Alamsyah (2010), dan Yoza, dkk. (2014), algoritma genetika merupakan bagian dari evolusi yang terbagi dalam beberapa tahap yaitu :

1. Inisialisasi
2. Evaluasi *Fitness*
3. Seleksi
4. *Crossover*
5. Mutasi

Berikut ini merupakan diagram alir dari AG yaitu



Gambar 3.3. Diagram Alir Algoritma Genetika
Sumber : Zukhri & Paputungan (2013)

Proses ini dimulai dari inisialisasi populasi, evaluasi *chromosome*, seleksi, *crossover* dan mutasi. Maximum generasi di gunakan untuk menghitung berapa banyak proses *iterasi*. Jika benar maka hasil pun di temukan. Dibawah ini merupakan *pseducode* AG menurut Zulfikar (2011) yaitu

```
Inisialisasi populasi awal, N kromosom
Loop untuk N generasi

        Loop untuk N kromosom
                Evaluasi kromosom
        End
        Elitisme
        Linear fitness ranking
        Loop sampai didapat N kromosom
baru
                Seleksi kromosom
                Pindah Silang
        End
```

Gambar 3.4. *Pseudocode* algoritma genetika
Sumber : Zulfikar (2011)

Menurut Murakhiro,dkk. (2007) pada Algoritma Genetika (GA) terdapat beberapa proses yaitu :

1. Proses Pengkodean (*Encoding*)

Pada proses pengkodean, gen dapat direpresentasikan dalam bentuk string bit, pohon, array bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.

2. Proses Seleksi

Seleksi adalah proses untuk menentukan individu mana saja yang akan dipilih untuk dilakukan rekombinasi dan bagaimana keturunan terbentuk dari individu-individu terpilih tersebut. Menurut Joni & Nurchayawati (2012), ada bermacam-macam teknik untuk melakukan proses seleksi pada suatu permasalahan. Teknik seleksi yang akan digunakan tergantung pada permasalahan yang akan diselesaikan. Ada beberapa metode seleksi dari induk, diantaranya adalah *Rank-basek Fitness Assignment,Roullete Wheel Selection, Stochastic Universal*

Sampling, Local Selection dan lain-lain. Proses penyeleksian yang digunakan disini adalah *Tournament Selection dan Elitism* .

*Tournament Selection s*eleksi turnamen bekerja dengan cara Seleksi turnamen bekerja dengan cara memilih beberapa jumlah t individu secara acak dari populasi dan mengopy individu terbaik dari grup ini ke dalam pupolasi,dan ulangi sebanyak N kali. Seringkali dalam turnamen yang dipegang hanya antara dua individu turnamen biner saja, tetapi umumnya memungkinkan untuk memegang sembarang ukuran kelompok t yang disebut dengan ukuran turnamen.

Metode *tournament Selection* ini mula-mula ditetapkan suatu nilai tour *t* untuk individu-individu yang dipilih secara random dari suatu populasi. Individu-individu yang terbaik dalam kelompok ini akan dipilih sebagai induk. Parameter yang digunakan pada metode ini adalah ukuran tour yang bernilai antara dua sampai N (jumlah individudalam suatu populasi) (Siami, 2012).

```
Binary_tournament_selection
(p,fv,n,N
S= new population of size
N;
For i=1 to n
A=rand (1,n);
B=rand (1,n-1);
If (b>=a) b++;
If (fv[a]>fv[b])
S[i]=P[a];
Else
S[i]=P[b];
Return S;
```

Gambar 3.5. *Pseducode tournament selection*
Sumber : Siami, (2012)

Perlu diperhatikan dalam proses ini ialah metode *elitism,* yang dilakukan dalam sekali seleksi untuk *update* generasi, biasanya digunakan *steady-state update*. Jadi tujuan utama dari *elitism* ini adalah untuk menjaga agar individu-individu yang bernilai *fitness* tertinggi tidak hilang selama proses evolusi, maka perlu dibuat kopiannya (Siami, 2012). Seleksi *elitism* bekerja dengan mengumpulkan semua individu dalam populasi (*parent*) dan *offspring* dalam satu

penampungan. Populasi terbaik dalam penampungan akan lolos untuk masuk dalam generasi selanjutnya. Hal ini menjamin individu yang terbaik akan selalu lolos (W.F, 2014).

Fungsi *elitism* menurut sebuah situs http://stackoverflow.com/questions/14622342/elitism-in-ga-should-i-let-the-elites-be-selected-as-parents di katakan bahwa setiap individu terbaik dijamin masuk ke generasi berikutnya dan umumnya tanpa mengalami mutasi. Mereka tetap dipilih untuk mempertahankan individu yang terbaik. Menurut Mitchell, 1997 dalam bukunya yang berjudul "*machine learning"* ditunjukkan bahwa *elitisme*, hanya menyalin N individu dalam generasi baru tanpa menerapkan setiap jenis perubahan. Dia tidak menghasilkan keturunan baru tetapi langsung di salin pada populasi baru.

Berikut ini dilampirkan potongan *source code menggunakan eltism* menurut (Jacobson, 2012)

```
// Don't mutate if this is the elite individual
and elitism is enabled
if (ga.elitism != true || index != fittestIndex) {
this.individuals[index].mutate();
}
}
```

Gambar 3.6 *Source Code* menggunakan *elitism*

Sumber : Jacobs (2012)

3. Proses Rekombinasi/ *crossover*

Rekombinasi adalah proses untuk menyilangkan dua kromosom sehingga membentuk kromosom baru yang harapannya lebih baik dari pada induknya. Rekombinasi dikenal juga dengan nama *crossover*. Tidak semua kromosom pada suatu populasi akan mengalami proses rekombinasi. Kemungkinan suatu kromosom mengalami proses rekombinasi didasarkan pada probabilitas *crossover*

yang telah ditentukan terlebih dahulu. Probabilitas *crossover* menyatakan peluang suatu kromosom akan mengalami *crossover* (Mutakhiroh, 2007).

Menurut penelitian Joni & Nurchayawati (2012) dalam penelitiannya *Penentuan Jarak Terpendek pada Jalur Distribusi Barang di Pulau Jawa dengan menggunakan Algoritma Genetika* digunakan teknik rekombinasi/*crossover* dengan nama proses *order crossover. Order crossover* ini diperkenalkan oleh Davis. Teknik ini diawali dengan membangkitkan dua bilangan acak. Kemudian gen yang berada pada *parent* kedua bilagan acakan akan disalin ke *offspringi* dengan posisi sama. Langkah berikutnya untuk mendapatkan *offspring* pertama adalah mengurutkan gen yang berada pada *parent* kedua dengan urutan gen yang berada pada posisi setelah bilangan acak kedua diikuti dengan gen yang berada pada posisi sebelum bilangan acak pertama dan diakhiri dengan gen yang berada pada posisi diantara ke dua bilangan acak.

Gen yang telah diurutkan tersebut dibandingkan dengan *offspring* pertama. Apabila gen tersebut ada pada *offspring* ke dua maka abaikan gen tersebut dari urutan itu. Kemudian masukan urutan yang baru saja di dapat pada *offspring* dengan cara memasukkan pada posisi sebelum bilangan acak pertama (Fachrurrazi, 2103).

Sebuah contoh diketahui *parent* adalah 678 ditambahkan ke dalam anak/keturunan. Untuk *parent* 9 yang tidak ikut pindah menduduki posisi pertama posisi berikutnya diikuti oleh 8 namun karena 8 sudah di pindahkan maka ini dilewati dan diisi oleh 5 dan seterusnya. Keturunan yang lain sampai semua terisi tanpa ada posisi yang hilang atau digandakan.

Perhatikan contoh perhitungan *crossover* pada gambar 3.7.

Parents

1	2	3	4	5	**6**	**7**	**8**	9

9	8	7	6	5	4	3	2	1

Offspring

					6	7	8	

9	**5**	**4**	**3**	**2**	6	7	8	1

Gambar 3.7. Contoh perhitungan *crossover*
Sumber : Jacobson (2012)

4. Proses Mutasi

Mutasi adalah proses penambahan nilai acak yang sangat kecil dengan probabilitas rendah pada variabel keturunan.Peluang mutasi didefinisikan sebagai persentasi dari jumlah total gen pada populasi yang mengalami mutasi. Peluang mutasi mengendalikan banyaknya gen baru yang akan dimunculkan untuk dievaluasi. Jika peluang mutasi terlalu kecil, banyak gen yang mungkin berguna tidak dievaluasi, tetapi bila peluang mutasi ini terlalu besar maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya dan algoritma juga akan kehilangan kemampuan untuk belajar dari *history* pencarian (Mutakhiroh, 2007).

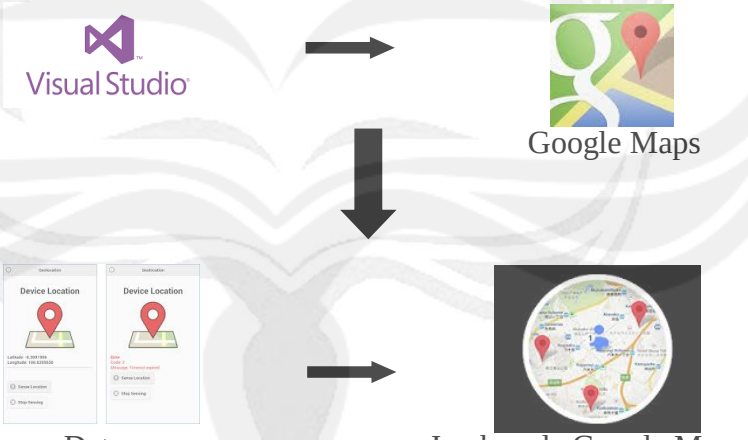
Teknik pertukaran yang digunakan dalam penelitian ini adalah *swapping mutation*. Menurut Hardianti & Purwanto (2013) dalam penelitiannya yang berjudul *Penerapan Algoritma Genetika dalam Penyelesain Travelling Salesman Problem With Predence Constraints (TSPPC)* mengemukakan bahwa cara kerja metode ini adalah sebagai berikut :

a. Membangkitkan bilangan acak sebanyak total bit (jumlah kromosom dikalikan dengan *pop_size*) untuk menetukan gen yang termutasi. Bilangan acak yang dibangkitan adalah real antara 0 sampai 1.
b. Mencari letak bilangan acak yang kurang *probabilitas* mutasi.
c. Menukarkan gen dengan bilangan acak kurang dari *probabilitas* mutasi dengan gen sesudahnya.

3.3 Google Maps Aplication Programming Interface (API)

Google Maps API merupakan aplikasi antarmuka yang dapat di akses lewat *Javascript* agar *google maps* dapat ditampilkan pada halaman web yang sedang dibangun. Berikut ini merupakan gambar diagram alir *request URL Google Maps* (Sanjaya , 2014).

Berikut ini menurut Pranata dkk (2009), adapun alur pemodelan yang digunakan dalam sisten ant colony optimization dapat dilihat pada gambar



Gambar 3.8. Alur pemodelan Google Maps
Sumber : Pranata dkk (2009)