BAB II

KAJIAN TEORI

Berikut diberikan beberapa teori pendukung untuk pembahasan selanjutnya.

2.1. Distribusi

Menurut Chopra dan Meindl (2010:86), distribusi adalah suatu kegiatan untuk memindahkan barang dari pihak *supplier* kepada pihak pelanggan dalam suatu *supply chain.* Distribusi merupakan suatu kunci dari keuntungan yang akan diperoleh perusahaan karena distribusi secara langsung akan mempengaruhi biaya dari *supply chain* dan kebutuhan pelanggan. Jaringan distribusi yang tepat dapat digunakan untuk mencapai berbagai macam tujuan dari *supply chain*, mulai dari biaya yang rendah sampai respon yang tinggi terhadap permintaan pelanggan.

Menurut Harry dan Syamsudin (2011), distribusi meliputi semua aspek dalam pengiriman barang kepada pelanggan. Distribusi juga merupakan bagian dari *material handling*, yaitu perpindahan material pada setiap saat dan setiap titik. Salah satu keputusan terpenting dalam hal pendistribusian adalah masalah penentuan jadwal serta rute pengiriman dari satu titik ke beberapa titik tujuan. Permasalahan tersebut memiliki beberapa tujuan yang ingin dicapai seperti tujuan untuk meminimumkan biaya pengiriman, meminimumkan waktu atau meminimumkan jarak tempuh. Salah satu dari tujuan tersebut bisa menjadi fungsi tujuan (*objective fuction)* dan yang lainnya menjadi kendala (*constraint*). Misal, fungsi tujuannya adalah meminimumkan biaya

pendistribusian, namun ada kendala *time window* dan kendala maksimum jarak tempuh tiap kendaraan, disamping kendala lain seperti kapasitas atau kendala lainnya.

2.2. Graf

Berikut akan diberikan pengertian graf dan jenis – jenis graf.

2.2.1. Pengertian Graf

Menurut Rinaldi (2010:356), sebuah graf $G = (V, E)$ terdiri atas sekumpulan objek $V = \{v_1, v_2, \dots, v_n\}$ yang disebut himpunan simpul, dan sebuah himpunan lain $E = \{e_1, e_2, \dots, e_n\}$ yang merupakan himpunan rusuk, sedemikian hingga setiap rusuk e_k dikaitkan dengan suatu pasangan tak terurut (v_i, v_j).

Banyaknya simpul dalam Graf G disebut *order* Graf G, sedangkan banyaknya rusuk dalam Graf G disebut *size* atau ukuran Graf G. Simpul-simpul pada graf dapat merepresentasikan sebuah objek sembarang seperti kota. Sedangkan rusuk-rusuk pada graf dapat menunjukkan hubungan sembarang seperti ruas jalan raya penghubung antar dua kota.

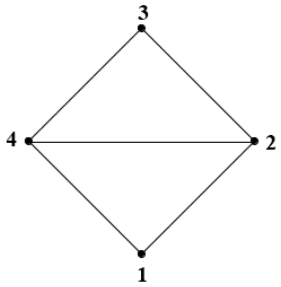
2.2.2. Jenis – Jenis Graf

Berdasarkan ada atau tidaknya *loop* atau rusuk ganda pada suatu graf, maka graf digolongkan menjadi dua jenis, yaitu sebagai berikut (Rinaldi, 2010:357):

1. Graf sederhana (*Simple Graph)*

Graf sederhana adalah graf yang tidak mengandung *loop* maupun rusuk berganda.

Contoh 2.1:



Gambar 2.1 Graf G_1

Graf G_1 pada Gambar 2.1 adalah graf sederhana dengan

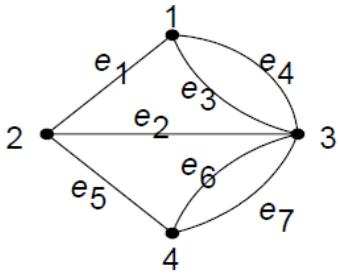
$$V = \{1, 2, 3, 4\}$$

$$E = \{(1, 2), (1, 4), (2, 3), (2, 4), (3, 4)\}$$

2. Graf tak-sederhana (Multi Graph)

Graf tak-sederhana adalah graf yang mengandung *loop* atau rusuk berganda.

Contoh 2.2:



Gambar 2.2 Graf G_2

Graf G_2 pada Gambar 2.2, adalah graf tak-sederhana dengan

$$V = \{1, 2, 3, 4\}$$

$$E = \{e_1 = (1, 2), e_2 = (2, 3), e_3 = (1, 3), e_4 = (1, 3),$$

$$e_5 = (2, 4), e_6 = (3, 4), e_7 = (3, 4)\}$$

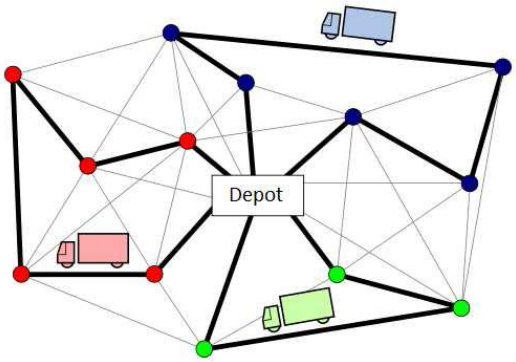
Rusuk $e_3 = (1,3)$ dan $e_4 = (1,3)$ pada Graf G_2, dinamakan rusuk berganda atau ruas sejajar (*multiple edges* atau *parralel edges*), karena kedua rusuk ini menghubungi dua buah titik yang sama, yaitu titik 1 dan titik 2.

2.3. *Vehicle Routing Problem*

Vehicle Routing Problem (VRP) didefinisikan sebagai masalah penentuan rute optimal untuk pendistribusian barang/jasa ke pelanggan-pelanggan dengan lokasi yang berbeda dengan permintaan yang sudah diketahui, dari satu atau lebih depot yang memenuhi beberapa kendala (Liong *et al*, 2008). Satu pelanggan tidak dapat dilayani oleh lebih dari satu kendaraan dalam satu periode permintaan sehingga akan ada k rute untuk memenuhi permintaan pelanggan karena keterbatasan kapasitas kendaraan. Hal tersebutlah yang menyebabkan VRP sering disebut sebagai permasalahan k-TSP (Chairul dkk, 2014).

Menurut Toth dan Vigo (2002) terdapat empat tujuan umum dari VRP, yaitu sebagai berikut:

1. Meminimumkan biaya transportasi, terkait dengan jarak dan biaya tetap yang berhubungan dengan kendaraan.
2. Meminimumkan jumlah kendaraan (atau pengemudi) yang dibutuhkan untuk melayani setiap pelanggan.
3. Menyeimbangkan rute, untuk waktu perjalanan dan muatan kendaraan.
4. Meminimumkan penalti akibat pelayanan yang kurang memuaskan dari pelanggan.



(sumber : Takes, 2010:12)

Gambar 2.3 Contoh Solusi VRP dengan 13 Pelanggan dan 3 Kendaraan

Terdapat beberapa variasi dalam permasalahan utama VRP (Toth dan Vigo, 2002) yaitu:

1. *Capacitated Vehicle Routing Problem* (CVRP)

   CVRP merupakan jenis VRP yang setiap kendaraannya memiliki kapasitas terbatas.

2. *Vehicle Routing Problem with Pick up and Delivery* (VRPPD)

   VRPPD merupakan jenis VRP dengan pelayanan jemput dan pelayanan antar dalam setiap permintaan pelanggan.

3. *Distance Constrained Vehicle Routing Problem* (DCVRP)

   DCVRP merupakan jenis VRP dengan kendala batasan panjang rute.

4. *Vehicle Routing Problem with Multiple Depot* (MDVRP)

   MDVRP merupakan jenis VRP yang memiliki banyak depot dalam melakukan pelayanan terhadap pelanggan.

5. *Split Delivery Vehicle Routing Problem* (SDVRP)

SDVRP merupakan jenis VRP dimana pelayanan terhadap pelanggan dilakukan dengan menggunakan kendaraan yang berbeda-beda.

6. *Vehicle Routing Problem with Time Windows* (VRPTW)

   VRPTW merupakan jenis VRP dengan kendala kapasitas kendaraan dan batasan waktu (*time windows*) pada setiap pelanggan dan depot.

2.4. *Capacitated Vehicle Routing Problem*

Capacitated Vehicle Routing Problem (CVRP) merupakan salah satu jenis permasalahan VRP. CVRP memiliki kendala berupa batasan kapasitas angkut kendaraan. Setiap kendaraan dengan kapasitas angkut tertentu harus melayani permintaan pelanggan tanpa melebihi kapasitas angkut kendaraan tersebut. Selain meminimumkan total jarak tempuh kendaraan, CVRP juga bertujuan untuk meminimumkan jumlah kendaraan yang digunakan dalam melayani pelanggan.

Menurut Kara *et al* (2004) masalah CVRP adalah masalah pengoptimalan jarak tempuh perjalanan kendaraan dalam pendistribusian barang dari depot ke sejumlah pelanggan-pelanggan sehingga menghasilkan rute dengan total jarak tempuh yang minimum. Penentuan rute kendaraan tersebut harus memperhatikan beberapa batasan yaitu setiap kendaraan harus memulai rute perjalanan dari depot dan setelah melayani sejumlah pelanggan juga harus kembali ke depot. Setiap pelanggan hanya dilayani tepat satu kali oleh satu kendaraan. Kendaraan-kendaraan tersebut memiliki kapasitas tertentu sehingga panjang rute yang dilalui oleh setiap kendaraan dalam melayani setiap pelanggan sesuai dengan kapasitasnya.

Pemodelan untuk CVRP memiliki parameter-parameter sebagai berikut :

n : banyaknya jumlah pelanggan,

q : kapasitas setiap kendaraan,

d_i : jumlah permintaan pelanggan i; dan

c_{ij} : jarak tempuh perjalanan dari pelanggan i ke pelanggan j.

Model matematika dari CVRP didefinisikan sebagai suatu graf $G = (V, E)$. Himpunan V terdiri atas gabungan himpunan pelanggan C dan depot, $V = \{0,1, \dots, n, n+1\}$. Himpunan C berupa pelanggan 1 sampai dengan n, $C = \{1,2 \dots, n\}$, dan depot dinyatakan dengan 0 dan $n + 1$. Jaringan jalan yang dilalui oleh kendaraan dinyatakan sebagai himpunan rusuk berarah E yaitu penghubung antar pelanggan, $E = \{(i,j) | i,j \in V, i \neq j\}$. Semua rute dimulai dan berakhir di depot. Himpunan kendaraan K merupakan kumpulan kendaraan yang homogen dengan kapasitas q. Setiap pelanggan i untuk setiap $i \in C$ memiliki permintaan d_i sehingga panjang rute dibatasi oleh kapasitas kendaraan. Setiap rusuk $(i,j) \in E$ memiliki jarak tempuh c_{ij}, dan juga bahwa $c_{ii} = c_{jj} = 0$.

Untuk setiap $(i,j) \in E, i \neq j \neq 0$ dan untuk setiap kendaraan k didefinisikan variabel:

$$x_{ijk} = \begin{cases} 1, \text{ jika terdapat perjalanan dari } i \text{ ke } j \text{ dengan kendaraan } k \\ 0, \text{ jika tidak terdapat perjalanan dari } i \text{ ke } j \text{ dengan kendaraan } k \end{cases}$$

Formula matematis untuk CVRP adalah sebagai berikut:

Meminimumkan $Z = \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ijk}$ (2.1)

dengan kendala

1. Setiap pelanggan dikunjungi tepat satu kali oleh suatu kendaraan:

$$\sum_{k\in K}\sum_{j\in V} x_{ijk} = 1, \qquad \forall i \in V \tag{2.2}$$

2. Total permintaan semua pelanggan dalam satu rute tidak melebihi kapasitas kendaraan:

$$\sum_{i\in V} d_i \sum_{j\in V} x_{ijk} \leq q, \qquad \forall k \in K \tag{2.3}$$

3. Setiap rute berawal dari depot 0:

$$\sum_{j\in V} x_{0jk} = 1, \qquad \forall k \in K \tag{2.4}$$

4. Setiap kendaraan yang mengunjungi satu pelanggan pasti akan meninggalkan pelanggan tersebut:

$$\sum_{i\in V} x_{ijk} - \sum_{j\in V} x_{ijk} = 0, \qquad \forall k \in K \tag{2.5}$$

5. Setiap rute berakhir di depot $n + 1$:

$$\sum_{i\in V} x_{i(n+1)k} = 1, \qquad \forall k \in K \tag{2.6}$$

6. Variabel x_{ijk} merupakan variable biner :

$$x_{ijk} \in \{0,1\}, \forall i, j \in V, \forall k \in K \tag{2.7}$$

Menurut Takes (2010:14), terdapat dua jenis CVRP. Jenis CVRP yang pertama adalah *Homogeneous* CVRP. *Homogeneous* CVRP adalah masalah CVRP dengan kapasitas tiap kendaraan sama. Jenis CVRP yang kedua adalah *Heterogeneous* CVRP. *Heterogeneous* CVRP adalah masalah CVRP yang kapasitas kendaraannya berbeda satu sama lain.

2.5. Algoritma Genetika

Berikut akan diberikan pengertian, istilah-istilah, komponen, dan parameter dari Algoritma Genetika.

2.5.1 Pengertian Algoritma Genetika

Algoritma Genetika merupakan suatu metode *heuristic* yang dikembangkan berdasarkan prinsip genetika dan proses seleksi alamiah Teori Evolusi Darwin. Metode optimasi dikembangkan oleh John Holland sekitar tahun 1960-an dan dipopulerkan oleh salah seorang mahasiswanya, David Goldberg, pada tahun 1980-an (Haupt dan Haupt, 2004:22).

Proses Algoritma Genetika secara umum untuk semua kasus adalah mendefinisikan individu, mendefinisikan nilai *fitness*, menentukan proses pembangkitan populasi awal, menentukan proses seleksi, menentukan proses perkawinan silang dan mutasi gen yang akan digunakan (Ahmad Basuki, 2003: 4).

2.5.2 Istilah – istilah dalam Algoritma Genetika

Berikut diberikan beberapa definisi dari istilah penting yang perlu diperhatikan dalam menyelesaikan masalah menggunakan Algoritma Genetika (Satriyanto, 2009:70):

1. Gen (*Genotype*) merupakan sebuah nilai yang menyatakan satuan dasar yang membentuk arti tertentu dalam satu kromosom. Gen dapat direpresentasikan dengan bilangan biner, *float*, integer, karakter, dan kombinatorial.
2. Kromosom merupakan gabungan gen-gen yang membentuk nilai tertentu.
3. Individu merupakan suatu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
4. Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi. Populasi terdiri dari sekumpulan kromosom.
5. Induk adalah kromosom yang akan dikenai operasi genetika (*crossover*).
6. *Crossover* atau pindah silang adalah operasi genetika yang mewakili proses perkembangbiakan antar individu.
7. *Offspring* yang juga dikenal sebagai keturunan atau anak adalah kromosom yang merupakan hasil dari pindah silang.
8. Mutasi merupakan operasi genetika yang mewakili proses mutasi dalam perjalanan hidup individu. Mutasi berperan menghasilkan perubahan acak dalam populasi, yang berguna untuk menambah variasi dari kromosom-kromosom dalam sebuah populasi.

9. Proses seleksi merupakan proses yang mewakili proses seleksi alam (*natural selection*) dari teori Darwin. Proses ini dilakukan untuk menentukan induk dari operasi pindah silang yang akan dilakukan untuk menghasilkan anak.

10. Nilai *fitness* merupakan penilaian yang menentukan bagus tidaknya sebuah kromosom.

11. Fungsi evaluasi adalah fungsi yang digunakan untuk menentukan nilai dari nilai *fitness*. Fungsi evaluasi ini merupakan sekumpulan kriteria-kriteria tertentu dari permasalahan yang ingin diselesaikan.

12. Generasi merupakan satuan dari populasi setelah mengalami operasi-operasi genetika, berkembang biak, dan menghasilkan keturunan. Kromosom-kromosom yang mempunyai nilai *fitness* yang rendah dan memiliki peringkat dibawah nilai minimal akan dihapus dari populasi pada akhir dari setiap generasi, untuk menjaga agar jumlah kromosom dalam populasi tetap konstan.

2.5.3 Komponen Algoritma Genetika

Algoritma Genetika memiliki beberapa komponen, diantaranya yaitu:

1. Penyandian Gen (Pengkodean)

Komponen ini merupakan proses penyandian gen dari kromosom. Gen merupakan bagian dari kromosom, satu gen akan mewakili satu variabel. Gen dapat direpesentasikan dalam bentuk bit, bilangan real, daftar aturan, elemen permutasi, elemen program atau representasi lainnya yang dapat diimplementasikan dalam operator genetika (Satriyanto, 2009:74).

Penelitian ini menggunakan teknik pengkodean permutasi pada representasi gen, yaitu tiap gen dalam kromosom merepresentasikan suatu urutan (Samuel, dkk, 2005).

Contoh 2.3: kromosom 1 = 2 3 4 5 1 6 7

Keterangan: kromosom 1 berisi urutan secara acak gen kesatu sampai ke tujuh. Gen direpresentasikan dengan sebuah bilangan dan bilangan-bilangan tersebut representasi dari masing-masing kota.

2. Membangkitkan Populasi Awal

Membangkitkan populasi awal dilakukan dengan membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Ukuran populasi tergantung pada masalah yang akan diselesaikan dan jenis operator genetika yang akan diterapkan. Setelah ukuran populasi ditentukan, kemudian dilakukan pembangkitan populasi awal menggunakan teknik tertentu (Sri Kusumadewi, 2003: 281).

Teknik dalam pembangkitan populasi awal ini ada beberapa cara, diantaranya adalah *random generator*, pendekatan tertentu, dan permutasi gen. Skripsi ini menggunakan teknik pembangkitan populasi berupa *random generator,* yaitu dengan melibatkan pembangkitan bilangan *random* untuk nilai setiap gen sesuai dengan representasi kromosom yang digunakan.

3. Menentukan Nilai *Fitness*

Fungsi yang digunakan untuk mengukur baik-tidaknya suatu individu sebagai solusi disebut dengan fungsi *fitness (fitness function)*. Nilai yang dihasilkan dari fungsi

tersebut menandakan seberapa optimal solusi yang diperoleh. Algoritma Genetika bertujuan mencari individu dengan nilai *fitness* tertinggi (Ahmad Basuki, 2003: 6).

Permasalahan CVRP bertujuan meminimalkan jarak, sehingga nilai *fitness* adalah inversi dari total jarak dari jalur yang didapatkan atau menggunakan rumus:

$$\text{Nilai } fitness = \frac{1}{x} \tag{2.8}$$

atau

$$\text{Nilai } fitness = 100.000 - x \tag{2.9}$$

dimana x adalah total jarak dari jalur yang didapatkan.

4. Seleksi

Seleksi bertujuan untuk memilih dua buah kromosom secara proporsional sesuai dengan nilai *fitness*-nya untuk dijadikan sebagai induk (Michalewicz, 1996:75). Proses pemilihan tersebut dapat dipilih berdasarkan probabilitas dari masing – masing individu. Probabilitas dari setiap individu tersebut ditentukan oleh nilai *fitness*nya masing – masing.

Menurut Kusumadewi (2003:105), terdapat beberapa metode yang dapat digunakan dalam melakukan seleksi, yaitu *rank-based fitness assignment*, *roulette wheel selection*, *stochastic universal sampling*, *local selection*, *truncation selection*, dan *tournament selection*. Skripsi ini menggunakan metode seleksi *Roulette Wheel.*

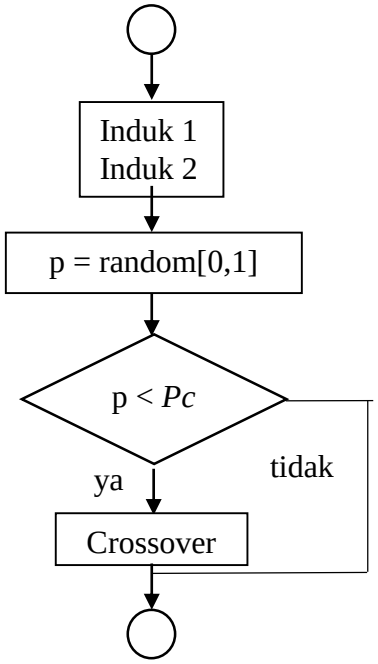
Metode seleksi *Roulette Wheel* merupakan metode yang sangat akurat dalam memilih kromosom untuk dijadikan sebagai induk. Metode ini seleksi yang menirukan permainan *Roulette Wheel* di mana masing-masing kromosom menempati potongan

lingkaran pada roda *roulette* secara proporsional sesuai dengan nilai *fitness*nya. Kromosom yang memiliki nilai *fitness* lebih besar menempati potongan lingkaran yang lebih besar dibandingkan kromosom bernilai *fitness* rendah. Sehingga semakin besar nilai *fitness* suatu kromosom maka semakin besar juga kesempatan kromosom tersebut untuk terpilih. Cara kerja metode seleksi ini yaitu dengan membuat interval nilai kumulatif dari nilai *fitness* masing masing kromosom dibagi total nilai *fitness* dari semua kromosom. Sebuah kromosom akan terpilih jika bilangan random yang dibangkitkan berada dalam interval kumulatifnya (Zainudin, 2014). Selain itu, metode seleksi *Roulette Wheel* ini juga mudah diimplementasikan dalam pemrograman.

5. *Crossover* (Pindah Silang)

Crossover atau pindah silang merupakan operator Algoritma Genetika yang bertujuan untuk membentuk kromosom baru dengan melibatkan dua induk yang telah terseleksi sebelumnya. Pindah silang akan menghasilkan sepasang anak baru dari dua induk. Setiap pasang induk akan dibangkitkan sebuah bilangan acak. Jika bilangan acak tersebut bernilai kurang dari *Probabilitas crossover* (Pc) antara 0,6 s/d 0,95 maka induk tersebut akan dikenai pindah silang. Jika pindah silang tidak dilakukan, maka nilai dari induk akan diturunkan sepenuhnya kepada anak (Michalewicz, 1996: 35).

Secara singkat, pindah silang adalah proses pertukaran gen yang bersesuaian dari dua induk untuk menghasilkan individu baru yang akan membawa sifat/gen induknya. Secara skematis, proses *crossover* dapat dilihat pada gambar 2.4.



Gambar 2.4 Skema alur proses *crossover*

Teknik *crossover* yang digunakan dalam skripsi ini adalah teknik *order crossover*. *Order crossover* (OX) diperkenalkan oleh Davis (Wira, 2010: 47). Teknik ini dijelaskan dalam contoh sebagai berikut:

Contoh 2.4:

Dari 2 induk diketahui:

Induk 1 = 1 2 3 | **4 5 6 7** |8 9

Induk 2 = 4 5 2 | **1 8 7 6** |9 3

Dibangkitkan 2 bilangan acak sebelum gen Induk 1 dan setelah gen Induk 1. Hal yang sama juga dilakukan untuk Induk 2. Didapatkan anak dengan gen yang sama:

Anak 1 = x x x | **4 5 6 7** |x x

Anak 2 = x x x | **1 8 7 6** |x x

Langkah berikutnya untuk mendapatkan Anak 1 adalah mengurutkan gen yang berada pada Induk 2 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 kedua bilangan acak.

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

Kemudian gen yang telah diurutkan tersebut dibandingkan dengan Anak 1. Apabila gen tersebut ada pada Anak 2 maka abaikan gen tersebut dari urutan itu. Kemudian urutan yang baru saja didapat dimasukkan pada anak dengan cara memasukkan urutan gen pada posisi setelah bilangan acak kedua terlebih dahulu dan sisanya dimasukkan pada posisi sebelum bilangan acak pertama. Begitu juga untuk menghasikan Anak 2. Anak 1 diperoleh:

Anak 1 = x x x | 4 5 6 7 |x x

Anak 1 = 2 1 8 | 4 5 6 7| 9 3

dengan jalan yang sama dibuat Anak 2 sehingga:

Anak 2 = x x x | 1 8 7 6 |x x

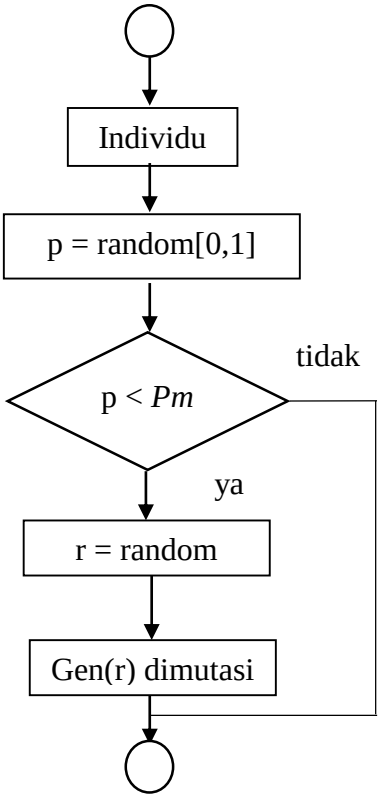
Anak 2 = 3 4 5 | 1 8 7 6 |9 2

6. Mutasi

Anak hasil proses pindah silang selanjutnya dilakukan proses mutasi. Variabel anak dimutasi dengan menambahkan nilai random yang sangat kecil, dengan probabilitas yang rendah. Probabilitas mutasi (Pm) didefinisikan sebagai persentasi dari jumlah total gen pada populasi yang mengalami mutasi. Pm mengendalikan

banyaknya gen baru yang akan dimunculkan untuk dievaluasi. Jika Pm terlalu kecil, banyak gen yang mungkin berguna tidak pernah dievaluasi. Tetapi jika Pm terlalu besar, maka akan terlalu banyak gangguan acak, sehingga anak akan kehilangan kemiripan dari induknya, dan juga algoritma akan kehilangan kemampuan untuk belajar dari pencarian sebelumnya (Sri Kusumadewi, 2003:296).

Ada beberapa teknik dalam melakukan mutasi. Teknik mutasi yang digunakan dalam skripsi ini adalah teknik *swapping mutation*. Teknik ini diawali dengan memilih dua bilangan acak kemudian gen yang berada pada posisi bilangan acak pertama ditukar dengan gen yang berada pada bilangan acak kedua dalam probabilitas tertentu (Suyanto, 2005: 65). Secara skematis, proses mutasi dapat dilihat pada gambar 2.5.

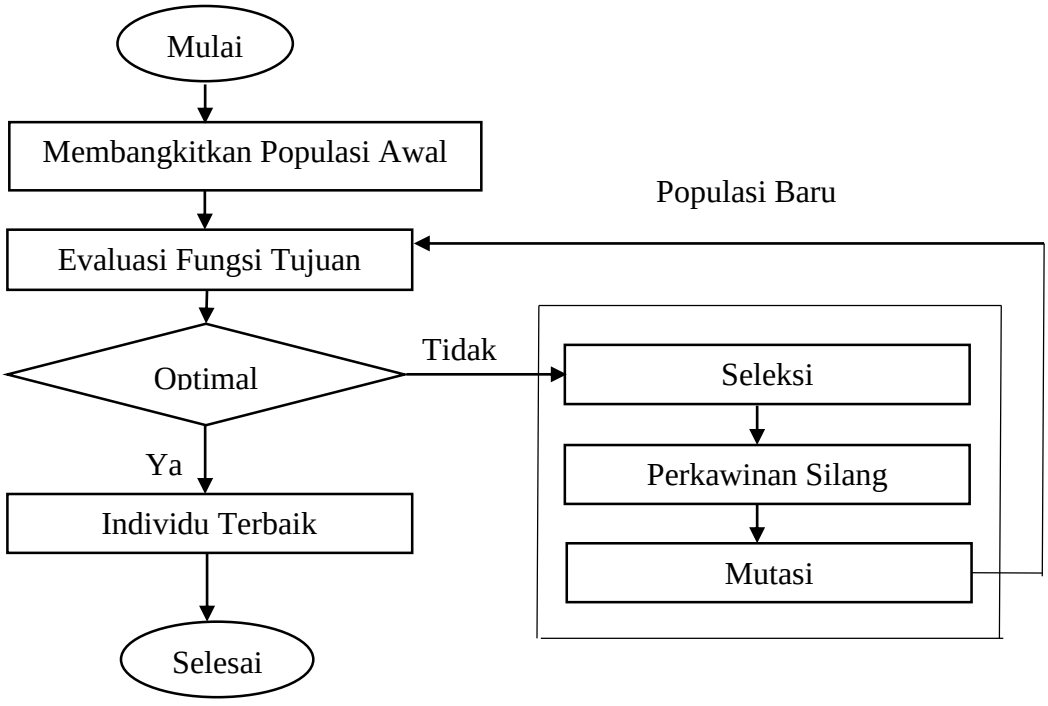


Gambar 2.5 Skema alur proses mutasi

7. *Elitism*

Elitism merupakan proses untuk menjaga agar individu bernilai *fitness* tertinggi tidak hilang selama evolusi. Proses evolusi merupakan proses Algoritma Genetika mulai dari pembentukan populasi awal hingga evaluasi nilai *fitness* dari populasi baru yang terbentuk. *Elitism* dilakukan karena proses seleksi dilakukan secara *random* sehingga tidak ada jaminan bahwa individu dengan nilai *fitness* tertinggi akan selalu dipilih. Walaupun individu bernilai *fitness* tertinggi terpilih, mungkin saja individu tersebut akan menurun nilai *fitness*nya karena proses pindah silang atau mutasi. Oleh sebab itu perlu dibuat satu atau beberapa *copy*nya untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi (Suyanto, 2005: 14).

Berikut diberikan skema alur Algoritma Genetika pada Gambar 2.6.



Gambar 2.6 Skema alur Algoritma Genetika

2.5.4 Parameter Algoritma Genetika

Proses Algoritma Genetika membutuhkan parameter – parameter yang ideal untuk digunakan selama pemrosesan Algoritma Genetika. Pemilihan parameter tersebut akan menentukan kinerja Algoritma Genetika dalam memecahkan masalah. Ada empat parameter dasar dari algoritma genetika, yaitu ukuran populasi, jumlah generasi, *probabilitas crossover (Pc)*, dan *probabilitas mutasi (Pm)*.

1. Ukuran populasi

Ukuran populasi menyatakan jumlah kromosom yang ada pada populasi. Ukuran populasi yang ideal berbeda-beda untuk setiap permasalahan. Menurut Obitko (1998), ukuran populasi yang ideal yaitu berkisar antara 20 sampai 30 kromosom.

2. Jumlah Generasi

Jumlah generasi menyatakan jumlah proses yang dijalani untuk suatu rangkaian proses Algoritma Genetika mulai dari proses seleksi, *crossover*, mutasi hingga pembentukan populasi baru. Jadi proses Algoritma Genetika akan berhenti jika jumlah generasi terpenuhi. terdapat solusi yang dicari yaitu individu dengan nilai *fitness* terbaik pada generasi terakhir.

3. Probabilitas *Crossover (Pc)*

Probabilitas *crossover* menyatakan seberapa sering proses *crossover* akan terjadi antara dua kromosom induk. Setiap individu dibangkitkan sebuah bilangan acak *p*, jika $p < Pc$ maka individu tersebut akan dilakukan *crossover* dan sebaliknya. Pemilihan nilai *Pc* yang ideal akan sangat mempengaruhi hasil dari proses *crossover* tersebut. Menurut Obitko (1998), probabilitas *crossover* yang ideal yaitu berkisar

antara 80% sampai 95%. Sedangkan menurut Michalewicz (1996:35), probabilitas *crossover* yang ideal yaitu berkisar antara 60% sampai 95%.

4. Probablitas Mutasi (*Pm*)

Probabliltas mutasi menyatakan seberapa sering bagian-bagian kromosom akan dimutasikan. Sama halnya seperti proses *crossover*, proses mutasi juga membangkitkan sebuah bilangan acak *p*, jika $p < Pm$ maka sebuah gen acak *r* dimutasi dan sebaliknya. Menurut Obitko (1998), probabilitas mutasi yang ideal yaitu berkisar antara 0,5% sampai 1%. Sedangkan menurut Suyanto (2009:14), probabilitas mutasi yang ideal adalah 1/jumlah gen.

Sri Kusumadewi (2003:283) mengemukakan ada beberapa rekomendasi nilai parameter yang bisa digunakan dalam proses Algoritma Genetika, yaitu sebagai berikut:

a. Untuk permasalahan yang memiliki kawasan solusi cukup besar, De Jong dalam Sri Kusumadewi (2003:283) merekomendasikan untuk nilai parameter kontrol:

   (ukupop*; Pc; Pm*) = (50;0.6;0.001)

b. Bila rata-rata *fitness* setiap generasi digunakan sebagai indikator, maka Grefenstette dalam Sri Kusumadewi (2003:283) merekomendasikan:

   (ukupop*; Pc; Pm*) = (30;0.95;0.01)

c. Bila *fitness* dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah:

   (ukupop*; Pc; Pm*) = (80;0.45;0.01)

d. Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan.

2.6. Metode *Nearest Neighbour*

Metode *Nearest Neighbour* merupakan metode yang digunakan untuk memecahkan masalah pemilihan rute dengan cara mencari jarak terpendek untuk menempuh lokasi pengiriman (Chairul, dkk. 2014). Prinsip dasar dari metode *Nearest Neighbour* yaitu membentuk rute perjalanan dengan memilih pelanggan yang terdekat dari lokasi awal. Metode *Nearest Neighbour* pertama kali diperkenalkan pada tahun 1983 dan merupakan metode yang sangat sederhana. Setiap iterasi dilakukan pencarian pelanggan terdekat dengan pelanggan yang terakhir untuk ditambahkan pada akhir rute tersebut. Rute baru dimulai dengan cara yang sama jika tidak terdapat posisi yang fisibel untuk menempatkan pelanggan baru karena kendala kapasitas atau *time* windows (Braysy & Gendreau, 2005).

Menurut Nissa dkk (2014), metode *Nearest Neighbour* merupakan suatu metode yang paling alami dalam menyelesaikan permasalahan VRP. Kendaraan bergerak menuju ke pelanggan-pelanggan terdekat yang belum dikunjungi dengan permintaan dari pelanggan tersebut tidak melebihi kapasitas kendaraan, tetapi apabila melebihi maka pengiriman dilakukan lebih dari satu kali namun setelah itu kendaraan menuju depot untuk *loading* kemudian menuju ke pelanggan terdekat selanjutnya.

Langkah-langkah metode *Nearest Neighbour* (Pop, 2011) adalah sebagai berikut:

1. Berawal dari depot, kemudian mencari pelanggan yang belum dikunjungi yang memiliki jarak terdekat dari depot sebagai lokasi pertama.

2. Ke pelanggan lain yang memiliki jarak terdekat dari pelanggan yang terpilih sebelumnya dan jumlah pengiriman tidak melebihi kapasitas kendaraan.

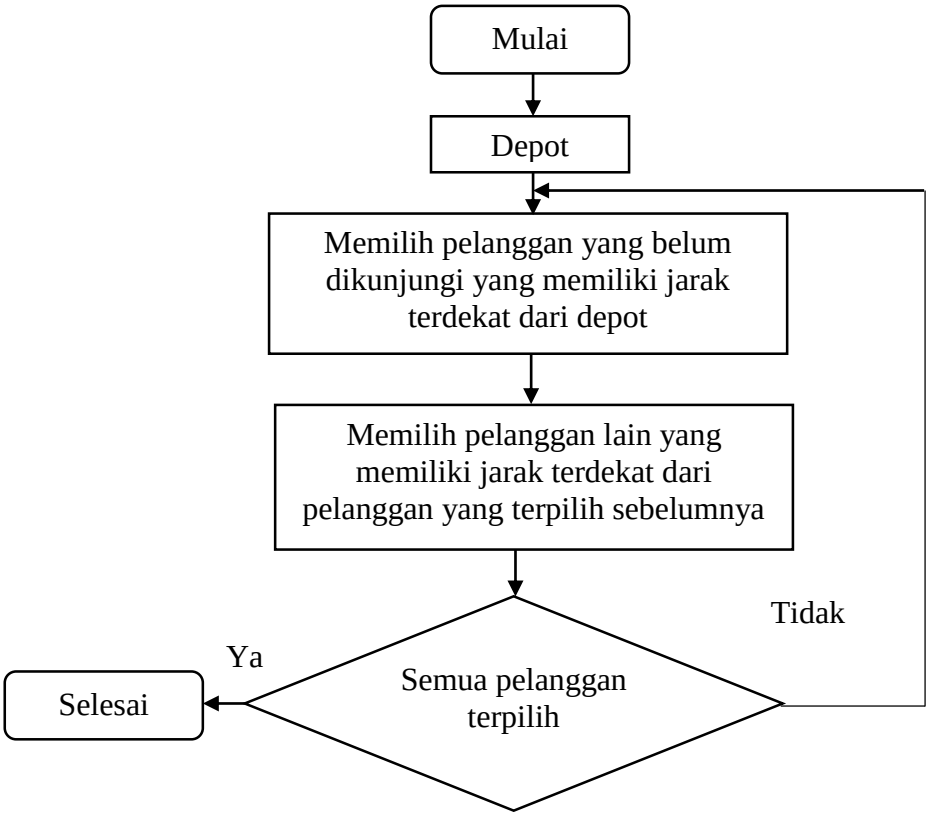
a. Apabila ada pelanggan yang terpilih sebagai pelanggan berikutnya dan terdapat sisa kapasitas kendaraan, kembali ke langkah (2).

b. Bila kendaraan tidak memiliki sisa kapasitas, kembali ke langkah (1).

c. Bila tidak ada lokasi yang terpilih karena jumlah pengiriman melebihi kapasitas kendaraan, maka kembali ke langkah (1). Dimulai lagi dari depot dan mengunjungi pelanggan yang belum dikunjungi yang memiliki jarak terdekat.

3. Bila semua pelanggan telah dikunjungi tepat satu kali maka algoritma berakhir.

Gambar 2.7 berikut menunjukkan skema alur Metode *Nearest Neighbour*.



Gambar 2.7. Diagram Alir Metode *Nearest Neighbour* Secara Umum