BAB II

KAJIAN TEORI

Secara umum, pada bab ini membahas mengenai kajian teori yang digunakan dalam penelitian yaitu teori graf, *vehicle routing problem* (VRP), *capacitated vehicle routing problem with time windows* (CVRPTW), algoritma genetika, dan variasi *crossover* dalam algoritma genetika yaitu *order crossover* dan *cycle crossover*.

A. Teori Graf

1. Definisi Graf

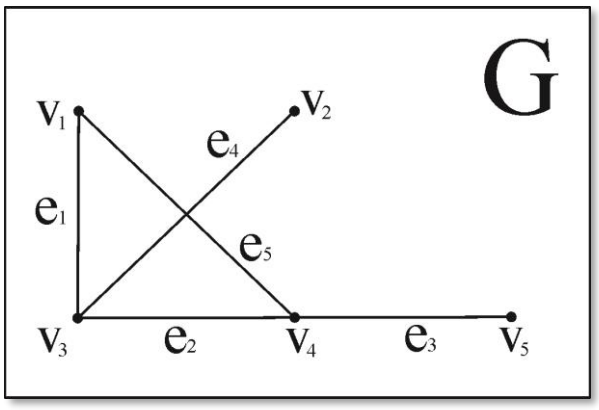
Graf adalah kumpulan titik yang dihubungkan satu sama lain melalui rusuk. Secara matematis, suatu graf G didefinisikan sebagai pasangan himpunan $G(V, E)$ dengan V adalah himpunan tidak kosong dari titik , $V(G) = \{v_1, v_2, v_3, .., v_n\}$, dan E adalah himpunan rusuk, $E(G) = \{e_1, e_2, e_3, \dots, e_n\}$, yang menghubungkan sepasang titik pada graf tersebut (Edgar G. Goodaire dan Michael M. Parmanter, 2002).

Graf G yang memuat himpunan titik Vdan himpunan rusuk Eseperti berikut ini.

$V = \{ v_1, v_2, v_3, v_4, v_5\}$

$E = \{e_1, e_2, e_3, e_4, e_5\}$

Graf Gtersebut dapat digambar sebagai berikut:



Gambar 2.1 Graf G

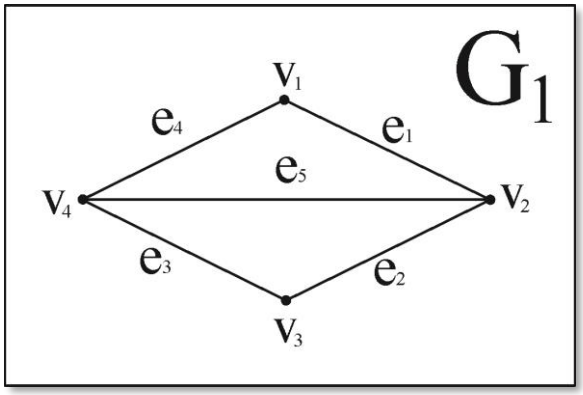
2. Jenis-Jenis Graf

Jenis-jenis graf dapat diklasifikasikan berdasarkan beberapa faktor-faktor sebagai berikut (Mardiyono, 2009):

a. Berdasarkan ada tidaknya *loop* atau rusuk ganda pada suatu graf, maka graf digolongkan menjadi dua jenis, yaitu:

1) Graf sederhana (*simple graph)*

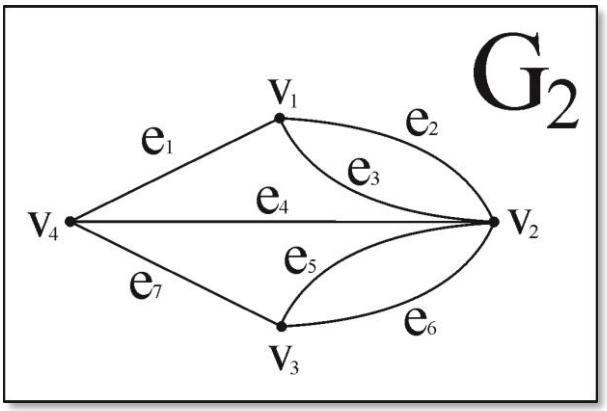
Graf sederhana yaitu graf yang tidak mengandung *loop* maupun rusuk ganda. Gambar G1 di bawah ini adalah contoh graf sederhana.



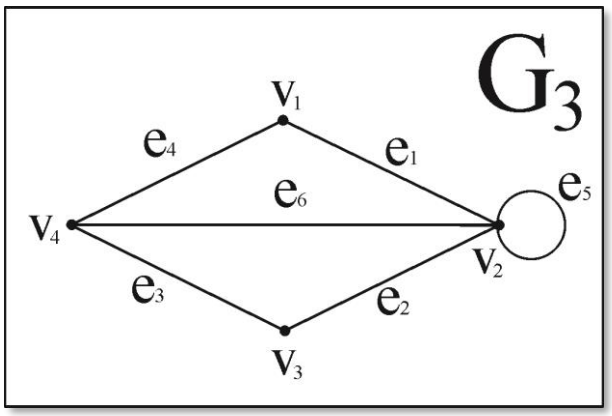
Gambar 2.2 Graf Sederhana G1

2) Graf tak-sederhana (*unsimple graph*)

Graf tak-sederhana yaitu graf yang mengandung *loop* atau rusuk ganda. Gambar Graf G2 di bawah ini adalah contoh graf tidak sederhana yang mengandung rusuk ganda dan Graf G3 adalah contoh graf tidak sederhana yang mengandung *loop*.



Gambar 2.3 Graf Tak Sederhana G2



Gambar 2.4 Graf Tak Sederhana G3

b. Berdasarkan jumlah titik pada suatu graf, maka secara umum graf dapat digolongkan menjadi dua jenis:

1) Graf berhingga (*limited graph*)

Graf berhingga adalah graf yang jumlah titiknya *n* berhingga.

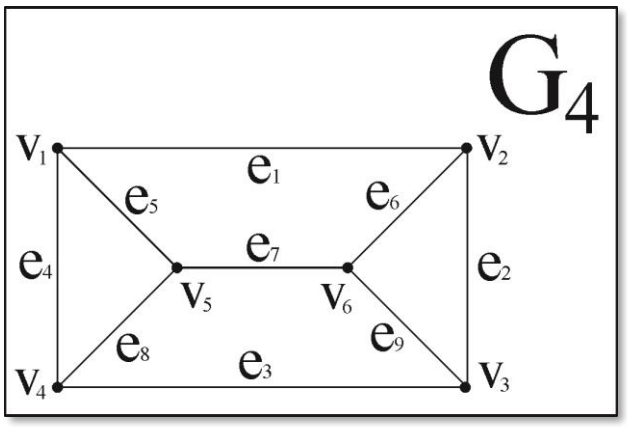
2) Graf tak berhingga (*unlimited graph*)

Graf tak behingga adalah graf yang jumlah titiknya *n* tidak berhingga banyaknya.

c. Berdasarkan orientasi arah pada rusuk, maka secara umum graf dibedakan atas dua jenis:

1) Graf tak berarah (*undirect grap*h)

Graf tak berarah adalah graf yang rusuknya tidak mempunyai orientasi arah. Urutan pasangan titik yang terhubung oleh rusuk tidak diperhatikan. Sehingga $(v_1, v_2) = (v_2, v_1)$ adalah rusuk yang sama. Graf tak berarah sering dipakai pada jaringan saluran telepon karena rusuk pada graf tak berarah menyatakan bahwa saluran telepon dapat beroperasi pada dua arah. Graf G4 di bawah ini merupakan contoh graf tak berarah.

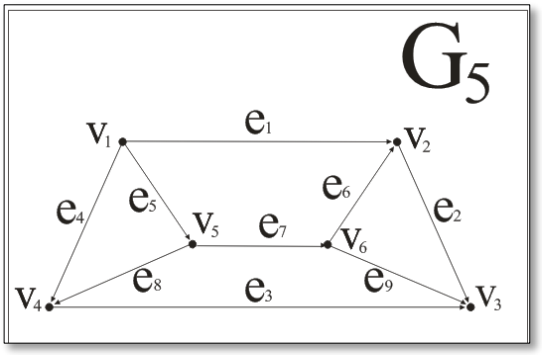


Gambar 2.5 Graf Tak Berarah G4

2) Graf berarah (*direct graph* atau *digraph*)

Graf berarah adalah graf yang setiap rusuknya diberikan orientasi arah. Pada graf berarah (v_1, v_2) dan (v_2, v_1) menyatakan dua buah rusuk yang berbeda, dalam arti kata bahwa $(v_1, v_2) \neq (v_2, v_1)$. Jadi untuk rusuk (v_1, v_2) titik

v_1 dinamakan titik asal dan titik v_2 dinamakan titik terminal atau titik tujuan. Graf berarah sering dipakai untuk menggambarkan aliran proses, peta lintas kota dan lain sebagainya. Sehingga pada graf berarah gelang atau *looping* diperbolehkan tetapi rusuk ganda tidak diperbolehkan. Berikut adalah gambar Graf G5 yang merupakan contoh graf berarah:



Gambar 2.6 Graf Berarah G5

3. Keterhubungan

Sebuah graf disebut terhubung jika untuk setiap dua titik terdapat rusuk yang menghubungkan kedua titik tersebut.

a. Jalan *(walk)*

Walk dengan panjang *k* pada sebuah graf G adalah rangkaian terurut dari *k* rusuk pada graf G dengan bentuk :

$$uv, vw, wx, \dots, yz \text{ ;}$$

Walk tersebut dinyatakan dengan $uv, vw, wx, \dots, yz$ atau dengan kata lain *walk* antara *u* sampai *z* (Robin J. Wilson & John J. Watkin, 1990 : 34).

b. Jejak (*Trail)*

Trail adalah *walk* dengan semua rusuk dalam barisan adalah berbeda (Munir, 2009 :370).

c. Lintasan (*Path)*

Path adalah sebuah *trail* tanpa titik berulang (Robin J. Wilson & John J. Watkin, 1990 : 35).

d. Sikel (*Cycle*)

Cycle adalah sebuah *trail* tertutup yang titik awal dan akhir merupakan titik yang sama.

f. Graf Hamilton

Sirkuit Hamilton adalah lintasan yang melalui tiap titik dalam graf tepat satu kali. Graf Hamilton adalah graf yang memiliki sirkuit Hamilton. Contoh sirkuit Hamilton adalah (v_1, v_2, v_3, v_4) pada G1. Contoh Graf Hamilton adalah G1.

B. *Vehicle Routing Problem* (VRP)

1. Pengertian VRP

Vehicle Routing Problem (VRP) dapat didefinisikan sebagai permasalahan mencari rute dengan biaya minimum dari suatu depot ke pelanggan yang letaknya tersebar dengan jumlah permintaan yang berbeda-beda. Rute dibuat sedemikian rupa sehingga setiap pelanggan dikunjungi hanya satu kali oleh satu kendaraan. Seluruh rute berawal di depot, dan jumlah permintaan dalam satu rute tidak boleh melebihi kapasitas kendaraan.

VRP diperkenalkan pertama kali oleh Dantzig dan Ramzer pada tahun 1959 memegang peranan penting dalam pengaturan distribusi dan menjadi salah satu masalah yang dipelajari secara luas.VRP merupakan perhitungan formulasi

dengan mempertimbangkan masalah jumlah kendaraan dan rute yang akan dilalui.

Tujuan yang ingin dicapai dalam VRP di antaranya (Toth &Vigo, 2002):

1. Meminimalkan ongkos perjalanan secara keseluruhan yang dipengaruhi oleh keseluruhan jarak yang ditempuh dan jumlah kendaraan yang digunakan.
2. Meminimalkan jumlah kendaraan yang digunakan untuk melayani semua pelanggan.
3. Menyeimbangkan rute.
4. Meminimalkan keluhan pelanggan.

2. Jenis-Jenis Vehicle Routing Problem (VRP)

Dalam perkembangannya, VRP memiliki beberapa variasi (Toth & Vigo, 2002), antara lain:

1. *Capacitated Vehicle Routing Problem (CVRP)*, dengan kendala yaitumasing-masing kendaraan memiliki kapasitas tertentu.
2. *Vehicle Routing Problem with Time Windows (VRPTW)*, merupakan jenis VRP dengan kendala interval waktu pelayanan.
3. *Multiple Depot Vehicle Routing Problem (MDVRP)*, merupakan jenis VRP yang memiliki banyak depot dalam melakukan pelayanan terhadap pelanggan.
4. *Vehicle Routing Problem with Pick-Up and Delivering (VRPPD)*, dengan kendala yaitu pelanggan dimungkinkan mengembalikan barang kepada agen asal. Pengiriman barang disuplai dari satu depot pada titik awal pengiriman, sementara *pick-up* muatan kemudian diambil untuk dikembalikan ke depot.

Karakteristik dari VRP –SPD adalah bahwa kendaraan yang digunakan pada suatu rute diisi oleh muatan barang yang dikirim dan muatan barang *pick-up*.

5. *Split Delivery Vehicle Routing Problem* (SDVRP), merupakan variasi dari CVRP dimana pelayanan terhadap pelanggan dilakukan dengan menggunakan kendaraan yang berbeda-beda.
6. *Periodic Vehicle Routing Problem (PVRP),* dengan faktor utama yaitu pengantaran hanya dilakukan di hari tertentu. Tujuan dari PVRP ini adalah untuk meminimalkan total jarak rute dan menyelesaikan permasalan penentuan jadwal pelayanan pelanggan.
7. *Vehicle Routing Problem with Multiple Products (VRPMP),* karakteristik dari VRP ini adalah permintaan pelanggan lebih dari satu jenis barang. Variasi dari semua VRP tersebut dapat digunakan sesuai dengan keadaan atau kondisi dari masalah yang dihadapi nantinya, tentunya dengan tujuan awal yang sama yaitu untuk meminimalkan total jarak tempuh untuk mendapatkan biaya transportasi yang minimum pula.
8. *Capacitated Vehicle Routing Problem with Time Windows* (CVRPTW), variasi dari VRP ini adalah VRP dengan memperhatikan kapasitas kendaraan dan waktu pelayanan.

C. *Capacitated Vehicle Routing Problem with Time Windows* (CVRPTW)

Capacitated vehicle routing problem with time windows (CVRPTW) adalah salah satu jenis VRP yang merupakan kombinasi dari bentuk umum

capacitated vehicle routing problem (CVRP) dan *vehicle routing problem with time windows* (VRPTW). CVRPTW bertujuan untuk membentuk rute optimal untuk memenuhi permintaan pelanggan yang dilakukan secara *delivery* dengan kendala kapasitas dan waktu pelayanan sehingga diperoleh waktu yang minimum.

CVRPTW termasuk dalam masalah VRP. Dalam CVRPTW, pelayanan pelanggan harus mulai dalam batasan waktu yang ditentukan dan kendaraan harus tiba di lokasi pelanggan selama waktu pelayanan. Jika kendaraan tiba sebelum pelanggan siap untuk memulai pelayanan, maka kendaraan harus menunggu terlebih dahulu (G. Confessore, G.Galiano. and G.Stecca, 2008).

Kendala pertama pada CVRPTW adalah kendala kapasitas. Kendala kapasitas yang dimaksud adalah bahwa setiap kendaraan memiliki kapasitas tertentu dan jika kapasitas kendaraan sudah penuh, maka kendaraan tersebut tidak dapat melayani pelanggan selanjutnya. Kendala berikutnya adalah kendala *time windows*. *Time windows* didefinisikan sebagai interval waktu pelayanan.

Masalah CVRPTW dapat direpresentasikan sebagai suatu graf berarah $G = (V, E)$ dengan $V = \{v_1, v_2, v_3, \dots, v_n\}$ adalah himpunan titik, v_i menyatakan depot yaitu tempat kendaraan memulai dan mengakhiri rute perjalanan. $E = \{(v_i, v_j) \,|v_i\,,\ v_j \in V, i \neq j\}$ adalah himpunan rusuk atau garis berarah yang menghubungkan dua titik yaitu ruas jalan penghubung antar pelanggan ataupun antara depot dengan pelanggan (Laporte, 1992).

Setiap titik $\{v_i \in V, i \neq 0\}$ memiliki permintaan sebesar q_i dengan q_i adalah bilangan bulat positif. Himpunan $K = \{k_1, k_2\,, \dots, k_n\}$ merupakan kumpulan kendaran yang heterogen, sehingga panjang setiap rute dibatasi oleh

kapasitas kendaraan. Setiap titik (v_i, v_j) memiliki waktu tempuh t_{ij} yaitu waktu tempuh dari titik i ke j. Waktu tempuh perjalanan ini diasumsikan simetrik yaitu $t_{ij} = t_{ji}$ dan $t_{ii} = 0$.

Dari permasalahan CVRPTW tersebut kemudian diformulasikan ke dalam bentuk model matematika dengan tujuan meminimumkan waktu distribusi kendaraan untuk melayani semua pelanggan, jika *z* adalah fungsi tujuan maka:

$$\min z = \sum_{i=1}^{n} \sum_{j=1}^{n} (t_{ij} \sum_{k=1}^{m} X_{ijk}) \qquad (2.1)$$

dengan variabel keputusan sebagai berikut:

1. Variabel $X_{ijk}, \forall i, j \in N, \forall k \in K, i \neq j$. Variabel X_{ijk} merepresentasikan ada atau tidaknya perjalanan dari pelanggan ke-i ke pelanggan ke-j oleh kendaraan ke-k.

$$X_{ijk} \begin{cases} 1, & jika\ ada\ perjalanan\ dari\ konsumen\ i \text{ ke pelanggan } j \text{ oleh kendaraan } k \\ 0, & jika\ tidak\ ada\ perjalanan\ dari\ konsumen\ i\ ke\ konsumen\ j\ oleh\ kendaraan\ k \end{cases}$$

2. Variabel T_{ik}, T_{0k}, $dan S_{ik} \forall\ i \in N, \forall k \in K$.

   Variabel$_k$ menyatakan waktu dimulainya pelayanan pada pelanggan ke-i oleh kendaraan ke-*k*. T_0 menyatakan waktu saat kendaraan ke-*k* meninggalkan depot dan kembali ke depot, dan S_{ik} menyatakan lamanya pelayanan di pelanggan ke-i.

3. Variabel Y_{ik} dan $q_j\ \forall i, j \in N, \forall k \in K$.

   Variabel Y_{ik} menyatakan kapasitas total dalam kendaraan ke-*k* setelah melayani pelanggan ke-*i*, sedangkan q_j menyatakan banyaknya permintaan pelanggan ke-j.

   Adapun kendala dari permasalahan CVRPTW adalah sebagai berikut:

1. Setiap pelanggan hanya dikunjungi tepat satu kali oleh kendaraan yang sama.

$$\sum_{i=1}^{25}\sum_{k=1}^{5} X_{i1k} = 1 \qquad (2.2)$$

.
.
.

$$\sum_{i=1}^{25}\sum_{k=1}^{5} X_{i25k} = 1$$

$$\sum_{j=1}^{25}\sum_{k=1}^{5} X_{1jk} = 1 \qquad (2.3)$$

.
.
.

$$\sum_{j=1}^{25}\sum_{k=1}^{5} X_{25jk=} = 1$$

2. Total jumlah permintaan pelanggan dalam satu rute tidak melebihi kapasitas kendaraan yang melayani rute tersebut. Jika ada lintasan dari *i* ke *j* dengan kendaraan *k*, maka

$$Y_{ik} + q_j = Y_{jk}, \forall i, k \in N, \forall k \in K \qquad (2.4)$$

$$Y_{jk} \leq Q, \forall j \in N, \forall k \in K \qquad (2.5)$$

3. Jika ada perjalanan dari pelanggan ke-*i* ke pelanggan ke-*j*, maka waktu memulai pelayanan di pelanggan ke-*j* lebih dari atau sama dengan waktu kendaraan ke-*k*untuk memulai pelayanan di pelanggan ke-*i* ditambah waktu pelayanan pelanggan ke-*i* dan ditambah waktu tempuh perjalanan dari pelanggan ke-*i* ke pelanggan ke-*j*. Jika ada lintasan dari *i* ke *j* dengan kendaraan *k*, maka

$$T_{ik} + s_{ik} + W_{tij} \leq T_{jk}, \forall i, j \in N, \forall k \in K \qquad (2.6)$$

1. Setiap pelanggan hanya dikunjungi tepat satu kali oleh kendaraan yang sama.

$$\sum_{i=1}^{25}\sum_{k=1}^{5} X_{i1k} = 1 \qquad (2.2)$$

.
.
.

$$\sum_{i=1}^{25}\sum_{k=1}^{5} X_{i25k} = 1$$

$$\sum_{j=1}^{25}\sum_{k=1}^{5} X_{1jk} = 1 \qquad (2.3)$$

.
.
.

$$\sum_{j=1}^{25}\sum_{k=1}^{5} X_{25jk=} = 1$$

2. Total jumlah permintaan pelanggan dalam satu rute tidak melebihi kapasitas kendaraan yang melayani rute tersebut. Jika ada lintasan dari *i* ke *j* dengan kendaraan *k*, maka

$$Y_{ik} + q_j = Y_{jk}, \forall i, k \in N, \forall k \in K \qquad (2.4)$$

$$Y_{jk} \leq Q, \forall j \in N, \forall k \in K \qquad (2.5)$$

3. Jika ada perjalanan dari pelanggan ke-*i* ke pelanggan ke-*j*, maka waktu memulai pelayanan di pelanggan ke-*j* lebih dari atau sama dengan waktu kendaraan ke-*k*untuk memulai pelayanan di pelanggan ke-*i* ditambah waktu pelayanan pelanggan ke-*i* dan ditambah waktu tempuh perjalanan dari pelanggan ke-*i* ke pelanggan ke-*j*. Jika ada lintasan dari *i* ke *j* dengan kendaraan *k*, maka

$$T_{ik} + s_{ik} + W_{tij} \leq T_{jk}, \forall i, j \in N, \forall k \in K \qquad (2.6)$$

4. Waktu kendaraan untuk memulai pelayanan di pelanggan ke-*i* harus berada pada selang waktu [a_i,b_i,]

$$a_i \leq T_{ik} \leq b_i, \forall\, i \in N, \forall\, k \in K \quad (2.7)$$

5. Setiap rute perjalan berawal dari depot dan berakhir di depot.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} X_{ijk} = 1, \ \forall\, i,j \in N, \forall k \in K \quad (2.8)$$

6. Kekontinuan rute, artinya kendaraan yang mengunjungi setiap pelanggan, setelah selesai melayani akan meninggalkan pelanggan tersebut.

$$\sum_{i=1}^{n} X_{ijk} - \sum_{i=1}^{n} X_{ijk} = 0, \forall\, i,j \in N, \forall k \in K \quad (2.9)$$

7. Variabel keputusan *Xijk* merupakan integer biner

$$X_{ijk} \in \{0,1\}, \forall i,j \in N, \forall k \in K \quad (2.10)$$

Berdasarkan fungsi tujuan dan kendala dari CVRPTW di atas, dapat dititikkan bahwa CVRPTW merupakan salah satu variasi dari VRP yang bertujuan untuk meminimalkan waktu distribusi dengan memperhatikan kapasitas kendaraan, total permintaan pelanggan, dan waktu pelayanan tiap pelayanan sehingga akan diperoleh rute yang optimum. Dalam kehidupan sehari-hari, sering dijumpai permasalahan mengenai CVRPTW. Misalnya, permasalahan pada pendistribusian barang dari depot ke sejumlah pelanggannya dengan batasan waktu pelayanan tertentu.

D. Algoritma Genetika

1. Definisi Algoritma Genetika

Algoritma Genetika pertama kali dikenalkan oleh John Holland pada tahun 1960. Bersama murid dan teman-temannya, John Holland mempublikasikan Algoritma Genetika dalam buku yang berjudul *Adaption of Natural and*

Artificial System pada tahun 1975 (Coley, 1999). Algoritma Genetika merupakan algoritma optimisasi yang terinspirasi oleh gen dan seleksi alam. Algoritma ini mengkodekan solusi-solusi yang mungkin ke dalam stuktur data dalam bentuk kromosom-kromosom dan mengaplikasikan operasi rekombinasi genetika ke stuktur data tersebut (Whitley, 2002).

Algoritma Genetika merupakan suatu metode heuristik untuk mencari solusi optimum dari suatu permasalahan dengan menggunakan mekanisme pencarian yang meniru proses evolusi biologis. Mekanisme yang digunakan merupakan kombinasi dari pencarian acak dan terstruktur. Algoritma ini sudah berhasil diterapkan dalam berbagai permasalahan kombinatorial, mulai dari *Traveling Salesman Problem* (*TSP*), *VRP*, dan penjadwalan produksi.

Dalam menyelesaikan penentuan kombinasi yang optimum, Algoritma Genetika berbeda dengan algoritma heuristik lainnya. Pada umunya, metode heuristik mencari solusi optimum dengan menyusun kombinasi secara bertahap berdasarkan kriteria pemilihan dan terminasi iterasi yang tertentu. Solusi yang didapatkan hanya satu macam solusi saja. Sebaliknya, Algoritma Genetika membuat suatu kode genetika dari kombinasi yang dimaksud, yang lebih dikenal sebagai istilah gen (*genotype*) yang selanjutnya disempurnakan dengan iterasi yang menyerupai proses alam dalam menurunkan sifat – sifat genetik. Algoritma genetika menggunakan mekanisme genetika yang ada pada proses alami dan sistem buatan. Istilah – istilah yang digunakan adalah gabungan dari dua disiplin ilmu, yaitu ilmu biologi dan ilmu komputer (John Willey & Sons, 1993).

Algoritma ini dimulai dengan pembentukan himpunan individu yang diwakili oleh kromosom. Himpunan kromosom ini disebut populasi awal. Populasi awal dapat dibentuk secara acak ataupun dengan metode heuristik (Ghoseiri & Ghamndpour, 2009). Sebelum membentuk populasi awal, dibutuhkan representasi solusi ke dalam kromosom.

Hal-hal yang terdapat dalam algoritma genetika adalah sebagai berikut (Weise, 2009):

1. Gen (*Genotype*) adalah sebuah nilai yang menyatakan satuan dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom.
2. *Allele* yaitu nilai dari sebuah gen, dapat berupa bilangan biner, float, integer, karakter dan kombinatorial.
3. Kromosom adalah gabungan gen – gen yang membentuk nilai tertentu.
4. Individu merupakan suatu nilai atau keadaan yang menyatakan salah satu solusi yang mungkin dari permasalahan yang diangkat.
5. Populasi merupakan sekumpulan individu yang akan diproses bersama dalam satu siklus proses evolusi. Populasi terdiri dari sekumpulan kromosom.
6. Induk adalah kromosom yang akan dikenai operasi genetik (*crossover*)
7. *Crossover* merupakan operasi genetik yang mewakili proses perkembangbiakan antar individu.
8. *Offspring* adalah kromosom yang merupakan hasil dari operasi genetik (*crossover*) dikenal sebagai keturunan atau anak.

9. Mutasi merupakan operasi genetik yang berperan menghasilkan perubahan acak dalam populasi, yang berguna untuk menambah variasi dari kromosom- kromosom dalam sebuah populasi.
10. Proses Seleksi merupakan proses yang mewakili proses seleksi alam (*natural selection*) dari teori Darwin. Proses ini dilakukan untuk menentukan induk dari operasi genetik (*crossover*) yang akan dilakukan untuk menghasilkan keturunan (*offspring*).
11. Nilai *fitness* merupakan penilaian yang menentukan bagus tidaknya sebuah kromosom.
12. Fungsi Evaluasi adalah fungsi yang digunakan untuk menentukan nilai *fitness*. Fungsi evaluasi ini merupakan sekumpulan kriteria-kriteria tertentu dari permasalahan yang ingin diselesaikan.
13. Generasi merupakan satuan dari populasi setelah mengalami operasi-operasi genetika, berkembang biak, dan menghasilkan keturunan. Pada akhir dari setiap generasi, untuk menjaga agar jumlah kromosom dalam populasi tetap konstan, kromosom–kromosom yang mempunyai nilai *fitness* yang rendah dan memiliki peringkat dibawah nilai minimal akan dihapus dari populasi.

Terdapat beberapa parameter yang digunakan dalam Algoritma Genetika. Parameter yang digunakan tersebut adalah (Sivanandam, 2008):

1. Jumlah Generasi

Merupakan jumlah perulangan (iterasi) dilakukannya rekombinasi dan seleksi. Jumlah generasi ini mempengaruhi kestabilan output dan lama iterasi (waktu proses Algoritma Genetika). Jumlah generasi yang besar dapat

mengarahkan ke arah solusi yang optimal, namun akan membutuhkan waktu *running* yang lama. Sedangkan jika jumlah generasinya terlalu sedikit maka solusi akan terjebak dalam lokal optimal.

2. Ukuran Populasi

Ukuran populasi mempengaruhi kinerja dan efektifitas dari Algoritma Genetika. Jika ukuran populasi kecil maka populasi tidak menyediakan cukup materi untuk mencakup ruang permasalahan, sehingga pada umumnya kinerja Algoritma Genetika menjadi buruk. Dalam hal ini dibutuhkan ruang yang lebih besar untuk mempresentasikan keseluruhan ruang permasalahan. Selain itu penggunaan populasi yang besar dapat mencegah terjadinya konvergensi pada wilayah lokal.

3. Probabilitas *Crossover* (Pc)

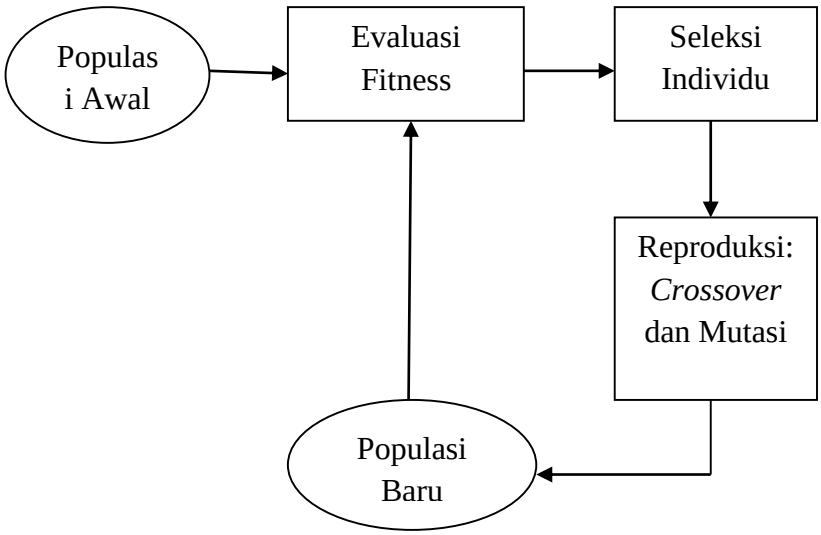
Probabilitas *crossover* ini digunakan untuk mengendalikan frekuensi operator *crossover*. Dalam hal ini, dalam populasi terdapat Pc dan ukuran populasi struktur yang akan melakukan *crossover*. Semakin besar nilai probabilitas *crossover* maka semakin cepat struktur baru diperkenalkan dalam populasi. Namun jika probabilitas *crossover* terlalu besar maka struktur dengan nilai fungsi obyektif yang baik dapat hilang dengan lebih cepat dari seleksi. Akibatnya populasi tidak dapat lagi meningkatkan nilai fungsi dari obyektifnya. Sebaliknya probabilitas *crossover* kecil akan menghalangi proses pencarian dalam proses Algoritma Genetika.Adapun mengenai probabilitas *crossover* yang baik, dari hasil penelitian yang dilakukan oleh Zbiniew Michalewics (1996) menyatakan bahwa probabilitas *crossover* yang baik adalah berada dalam interval [0.65-1].

4. Probabilitas Mutasi (Pm)

Mutasi digunakan untuk meningkatkan variasi populasi.Probabilitas mutasi ini digunakan untuk menentukan tingkat mutasi yang terjadi. Probabilitas mutasi yang rendah akan menyebabkan gen – gen yang berpotensi tidak dicoba, dan sebaliknya, tingkat mutasi yang tinggi akan menyebabkan keturunan semakin mirip dengan induknya. Adapun mengenai probabilitas mutasi yang baik, dari hasil penelitian yang dilakukan oleh Zbiniew Michalewics (1996) menyatakan bahwa probabilitas mutasi yang baik adalah berada dalam interval [0.01 – 0.3].

2. Skema Algoritma Genetika

Skema algoritma genetika pertama kali dikemukakan oleh David Goldberg (1989), dengan skema tersebut dapat dilihat pada gambar beikut:



Gambar 2.7 Skema Algoritma Genetika oleh David Goldberg (1989)

3. Komponen Algoritma Genetika

a. Teknik Penyandian (Pengkodean)

Teknik penyandian adalah proses penyandian gen dari kromosom. Gen merupakan bagian dari kromosom, satu gen biasanya 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).

Terdapat beberapa teknik pengkodean dalam algoritma genetika diantaranya pengkodean biner, pengkodean permutasi, pengkodean nilai dan pengkodean pohon (Anwar dan Yuliani, 2005). Pada penelitian ini, representasi gen menggunakan teknik pengkodean permutasi.Dalam pengkodean ini, tiap gen dalam kromosom merepresentasikan suatu urutan (Anwar dan Yuliani, 2005).

Contoh 2.1 kromosom 1 = 3 5 1 6 4 2

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 pelanggan dan depot.

b. Membangitkan Populasi Awal *(Spanning)*

Membangkitkan populasi awal adalah membangkitkan sejumlah individu secara acak atau melalui prosedur tertentu. Ukuran populasi tergantung pada masalah yang akan dipecahkan dan jenis operator genetika yang akan diimplementasikan. Setelah ukuran populasi ditentukan, kemudian dilakukan inisialisasi terhadap kromosom yang terdapat pada populasi tersebut. Inisialisasi

kromosom dilakukan secara acak, namun demikian harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada (Kusumadewi, 2003: 282).

Terdapat berbagai teknik dalam pembangkitan populasi awal diantaranya adalah *random generator*, pendekatan tertentu, dan permutasi gen. Pada penelitian ini pembangkitan populasi awal dilakukan dengan menggunakan random generator. Inti dari cara ini adalah melibatkan pembangkitan bilangan random dalam interval (0,1) untuk setiap nilai gen sesuai dengan representasi kromosom yang digunakan.

c. Mengevaluasi Nilai *Fitness* (*Fitness Value*)

Mengevaluasi nilai *fitness* berfungsi untuk mengukur kualitas dari sebuah solusi dan memungkinkan tiap solusi untuk dibandingkan (Michalewicz, 1996: 72). Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran baik tidaknya individu tersebut. Di dalam evolusi alam, individu yang bernilai *fitness* tinggi yang akan bertahan hidup, sedangkan individu yang bernilai *fitness* rendah akan mati (D.E.Goldberg, 1989). Pada masalah optimasi, fungsi *fitness* yang digunakan adalah

$$F = \frac{1}{p} \tag{2.11}$$

dengan p merupakan nilai dari individu, yang artinya semakin kecil nilai p, maka semakin besar nilai *fitness*nya. Tetapi hal ini akan menjadi masalah jika p bernilai 0, yang mengakibatkan F bisa bernilai tak hingga. Untuk mengatasinya, p perlu ditambah sebuah bilangan sangat kecil sehingga nilai *fitness*nya menjadi

$$F = \frac{1}{(p+a)}, \tag{2.12}$$

dengan a adalah bilangan yang dianggap sangat kecil (konstanta) dan bervariasi sesuai dengan masalah yang akan diselesaikan (Suyanto, 2005:10).

d. Seleksi (*Selection*)

Seleksi merupakan pemilihan dua buah kromosom untuk dijadikan sebagai induk yang dilakukan secara proporsional sesuai dengan dengan nilai *fitness*-nya (Michalewicz, 1996: 75). Masing-masing individu yang diseleksi akan diberikan probabilitas reproduksi tergantung dari nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam seleksi tersebut. Nilai *fitness* inilah yang nantinya akan digunakan pada tahap seleksi berikutnya.

Terdapat beberapa metode seleksi menurut Kusumadewi (2003:105), yaitu *rank-based fitness assignment*, *roulette wheel selection*, s*tochastic universal sampling*, seleksi lokal (l*ocal selection*), seleksi dengan pemotongan (*truncation selection*) dan seleksi dengan turnamen (*tournament selection*). Skripsi ini menggunakan metode *Roulette Wheel Selection*.

Cara kerja metode *Roulette Wheel Selection* adalah sebagai berikut

a. Dihitung nilai *fitness* dari masing-masing individu (F_i, dimana i adalah individu ke-1 s/d ke-n)
b. Dihitung total *fitness* semua individu
c. Dihitung probabilitas masing-masing individu
d. Dari probabilitas tersebut, dihitung jatah masing-masing individu pada angka 1 sampai 100
e. Dibangkitkan bilangan random antara 1 sampai 100

f. Dari bilangan random yang dihasilkan, ditentukan individu mana yang terpilih dalam proses seleksi

IlustrasiSeleksi dengan Mesin *Roulette*

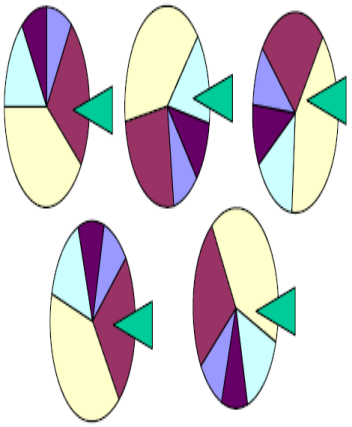
Individu 1: fitness =10%
Individu 2: fitness =25%
Individu 3: fitness =40%
Individu 4: fitness =15%
Individu 5: fitness =10%

Jatah untuk individu 1=1 s.d 10
Jatah untuk individu 2=11 s.d 35
Jatah untuk individu 3=36 s.d 75
Jatah untuk individu 4=76 s.d 90
Jatah untuk individu 5=91 s.d 100

Dibangkitkan Bilangan Random antara 1 s.d 100 sebanyak 5 kali

Individu Terpilih

Random 30= Individu 2
Random 88= Individu 4
Random 64= Individu 3
Random 18= Individu 2
Random 44= Individu 3



e. Pindah Silang (*Crossover*)

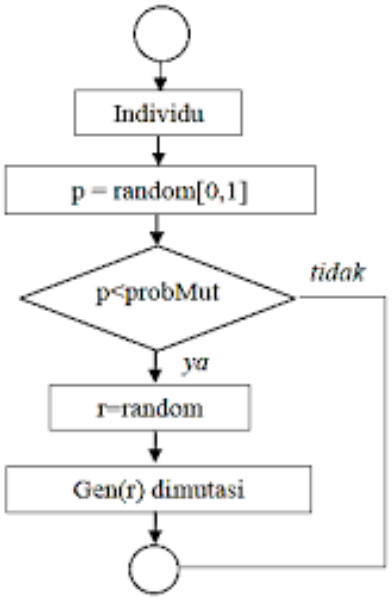
Pindah Silang (*crossover*) adalah operator dari algoritma genetika yang melibatkan dua induk untuk membentuk kromosom baru. Pindah silang

menghasilkan keturunan baru dalam ruang pencarian yang siap diuji. Operasi ini tidak selalu dilakukan pada setiap individu yang ada. Individu dipilih secara acak untuk dilakukan *crossover* dengan Pc (*Probabilitas Crossover*) antara 0,6 s/d 0,95. Jika pindah silang tidak dilakukan, maka nilai dari induk akan diturunkan kepada keturunan (Michalewicz, 1996: 78).

Prinsip dari pindah silang adalah melakukan operasi (pertukaran aritmatika) pada gen yang bersesuaian dari dua induk untuk menghasilkan individu baru. Proses *crossover* dilakukan pada setiap individu dengan probabilitas *crossover* yang ditentukan.

e. Mutasi (*Mutation*)

Mutasi merupakan proses untuk mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operasi mutasi yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang dengan *fitness* yang lebih baik, dan lama-kelamaan menuju solusi optimum yang diinginkan.Akan tetapi, untuk mencapai hal ini, penekanan *selektif* juga memegang peranan yang penting. Jika dalam proses pemilihan kromosom-kromosom cenderung terus pada kromosom yang memiliki *fitness* yang tinggi saja, konvergensi prematur akan sangat mudah terjadi (Murniati, 2009: 24). Secara skematis proses mutasi dapat digambarkan sebagai berikut.



Gambar 2.8 Skematika Proses Mutasi

Dari gambar 2.8 di atas, jika p merupakan bilangan random yang dibangkitkan kurang dari probabilitas mutasi (probMut) maka individu hasil *crossover* dilakukan proses mutasi Sedangkan jika bilangan p yang dibangkitkan lebih dari atau sama dengan probMut, maka individu hasil *crossover* tidak dilakukan proses mutasi. Teknik *swapping mutation* diawali dengan memilih dua bilangan acak kemudian gen yang berada pada porusuk bilangan acak pertama ditukar dengan gen yang berada pada bilangan acak kedua dalam probabilitas tertentu (Suyanto, 2005: 57).

Contoh

Sebelum

1	2	3	4	5	6

Sesudah

1	4	3	2	5	6

f. Elitism

Elitism merupakan proses untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi (Kusumadewi, 2003: 112). Proses seleksi dilakukan secara random sehingga tidak ada jaminan bahwa suatu indvidu yang bernilai *fitness* tertinggi akan selalu terpilih. Walaupun individu bernilai *fitness* tertinggi terpilih, mungkin saja individu tersebut akan rusak (nilai *fitness*-nya menurun) karena proses pindah silang. Oleh karena itu, untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, maka perlu dibuat satu atau lebih. Proses *elitism* dilakukan dengan menduplikat individu dengan nilai *fitness* terbaik untuk dijadikan individu pertama pada generasi berikutnya.

g. Pembentukan Populasi Baru

Proses membangkitkan populasi baru bertujuan untuk membentuk populasi baru yang berbeda dengan populasi awal. Pembentukan populasi baru ini didasarkan pada keturunan-keturunan baru hasil mutasi ditambah dengan individu terbaik setelah dipertahankan dengan proses *elitism*. Setelah populasi baru terbentuk, kemudian mengulangi langkah-langkah evaluasi nilai fitness, proses seleksi, proses pindah silang, proses mutasi pada populasi baru untuk membentuk populasi baru selanjutnya.

E. *Crossover* dalam CVRPTW

Crossover merupakan bagian terpenting dalam algoritma genetika, karena di sini ditentukan bagaimana membentuk generasi yang baru. *Crossover* (pindah silang) adalah metode mengawinkan dua kromosom induk (orang tua) dengan

tujuan untuk mendapatkan anak yang lebih baik (Fanggidae, A & Lado, F.R, 2015:9). *Crossover* bekerja dengan membangkitkan *offspring* baru dengan mengganti sebagian informasi dari induk (Mahmudy, 2013). Beberapa metode *crossover* yang akan digunakan untuk menyelesaikan persoalan CVRPTW pada peneltian ini, sebagai berikut:

1. *Order Crossover* (OX)

Order Crossover (OX) membutuhkan urutan sejumlah gen dari suatu kromosom yang akan dilakukan *crossover,* sehingga dilakukan penentuan posisi awal dan akhir gen dari suatu kromosom. Algoritma dari OX (Kumar, dkk., 2012) adalah sebagai berikut:

a. Tentukan 2 bilangan secara acak (bilangan bulat dari 1 sampai panjang gen dari suatu kromosom) yang akan dijadikan porusuk awal dan akhir gen dari suatu kromosom yang akan mengalami *crossover,* tandai gen-gen tersebut.

1) Bentuk anak 1 dengan:

a) Kosongkan gen-gen pada induk 1 yang termasuk dalam gen-gen yang ditandai pada induk 2.

b) Geser ke kanan *allele-allele* pada induk 1 sampai gen-gen yang ditandai pada induk 1 kosong.

c) Isi gen-gen yang ditandai (yang telah kosong) pada induk 1 dengan *allele-allele* yang telah ditandai pada induk 2.

2) Bentuk anak 2 dengan:

a) Kosongkan gen-gen pada induk 2 yang termasuk dalam gen-gen yang ditandai pada induk 1.

b) Geser ke kanan *allele-allele* pada induk 2 sampai gen-gen yang ditandai pada induk 2 kosong.

c) Isi gen-gen yang ditandai (yang telah kosong) pada induk 2 dengan *allele-allele* yang telah ditandai pada induk 1.

Contoh ilustrasi *order crossover* :

Dipunyai 2 bilangan acak, yaitu 2 dan 3.

Posisi awal gen yang akan di*crossover* = 2

Posisi akhir gen yang akan di*crossover* = 3

Induk 1 :

A	B	C	D	E

Induk 2 :

C	D	A	E	B

Anak 1 :

	B	C		E

E			B	C

E	D	A	B	C

Anak 2:

	D	A	E	

E			D	A

E	B	C	D	A

2. *Cycle Crossover* (CX)

Pada metode ini akan dilakukan *cycle* antara dua induk, yang dimulai dari porusuk awal gen kromosom induk 1 dan akan berhenti pada gen yang tidak dilanjutkan *cycle* nya.

Algoritma dari CX (Kumar, dkk., 2012) adalah sebagai berikut:

a. Tentukan pola *cycle* dari duainduk, yang dimulai dari porusuk awal gen kromosom induk 1.

1) Bentuk anak 1 dengan:

a) Tandai *allele-allele* pada induk 1 yang termasuk dalam pola *cycle* (hasil dari langkah 1).

b) Ambil secara berurutan *allele-allele* yang tidak ditandai pada induk *2* (yang tidak termasuk dalam *cycle*) untuk diisi pada gen yang masih kosong pada induk 1.

2) Bentuk anak 2 dengan:

a) Tandai *allele-allele* pada induk 2 yang termasuk dalam pola *cycle* (hasil dari langkah 1).

b) Ambil secara berurutan *allele-allele* yang tidak ditandai pada induk 1(yang tidak termasuk dalam *cycle*) untuk diisi pada gen yang masih kosong pada induk 2.

Contoh ilustrasi *cycle crossover*:

Induk1

B	E	C	D	A

Induk 2:

C	D	A	E	B

Pola *cycle*: B → C → A → B

Anak 1:

B		C		A

B	D	C	E	A

Anak 2:

C		A		B

C	E	A	D	B