BAB III
MODEL CTSP DAN PENYELESAIANNYA MENGGUNAKAN ALGORITMA GENETIK *HILL-CLIMBING*

3.1 Deskripsi Masalah

Penelitian ini membahas penyelesaian masalah optimisasi *Colored Traveling Salesman Problem* dengan menggunakan Algoritma Genetika *Hill-Climbing*. CTSP adalah perkembangan dari permasalahan MTSP yaitu menentukan rute terpendek dari beberapa pekerja. Pada CTSP wilayah kerja terbagi menjadi 2 yaitu wilayah umum dan wilayah pribadi. Wilayah umum merupakan wilayah yang dapat dikunjungi oleh seluruh pekerja dan wilayah pribadi merupakan wilayah yang hanya dapat dikunjungi oleh pekerja yang ditugaskan. Tujuan dari penyelesaian CTSP pada penelitian ini adalah untuk menentukan rute dengan jarak terpendek dari masing-masing kendaraan dan tidak boleh terjadi tumpang tindih pada suatu titik atau kota jika pekerja berangkat dari depot dan kembali ke depot. Setiap pekerja dapat mengunjungi kota di wilayah umum dan diharuskan mengunjungi setiap kota pada wilayah pribadi yang telah ditentukan.

Pada bab ini dikaji model dari permasalahan CTSP dan kendala-kendalanya dengan tujuan meminimumkan jarak. Selanjutnya dijelaskan cara kerja Algoritma Genetika *Hill-Climbing* dan metode-metode yang digunakan dalam Algoritma Genetika untuk menyelesaikan permasalahan CTSP.

3.2 Model *Colored Traveling Salesman Problem* (CTSP)

Pada bagian ini dikaji model optimisasi *Colored Traveling Salesman Problem* yang sebelumnya telah dirumuskan oleh Li (2014). Asumsi-asumsi yang digunakan pada pemodelan ini adalah sebagai berikut

1. Hanya terdapat satu depot.
2. Setiap kota dikunjungi tepat satu kali oleh satu pekerja.
3. Setiap rute dilalui oleh satu pekerja.
4. Banyak kota yang dikunjungi oleh setiap pekerja berjumlah sama.

5. Setiap rute jalan dapat dilewati.
6. Jarak lokasi dari i ke j sama dengan j ke i

Selanjutnya didefinisikan himpunan, parameter dan variabel keputusan yang akan digunakan pada model ini. Berikut himpunan yang digunakan dalam model ini:

V : himpunan kota

E : himpunan pekerja

P_k : himpunan kota yang merupakan wilayah pribadi, dimana $P_k \subset V$

U : himpunan kota yang merupakan wilayah umum, dimana $U \in V \setminus P_k$

Adapun parameter yang digunakan sebagai berikut:

C_{ij} : jarak dari kota i ke kota j, dimana $i, j \in V$

n : banyak kota

m : banyak pekerja

Variabel keputusan model CTSP didefinisikan untuk menentukan ada tidaknya perjalanan dari kota i ke kota j oleh pekerja k, dimana setiap rute harus berangkat dari depot yang selanjutnya direpresentasikan oleh bilangan 0. Variabel keputusanya yaitu x_{ijk} dimana variabel x_{ijk} bernilai 1 jika terdapat perjalanan dari kota i ke kota j yaitu jarak dari kota i ke kota j optimal atau bernilai paling kecil, dan bernilai 0 jika sebaliknya.

Selain himpunan, parameter dan variabel keputusan, terdapat fungsi tujuan dan kendala dari model CTSP. Fungsi tujuan dari CTSP adalah meminimumkan total jarak perjalanan setiap rute. Berdasarkan Li (2014), fungsi tujuan tersebut dituliskan sebagai:

$$f = \sum_{k=1}^{m} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} C_{ij} x_{ijk}$$

Selanjutnya didefiniskan kendala-kendala model CTSP sebagai berikut:

1. Setiap pekerja berangkat dari depot dan kembali pada depot. Kendala ini diekspresikan sebagai:

$$\sum_{j=1}^{n-1} x_{0jk} = 1 \, , k \in Z_m$$

$$\sum_{i=1}^{n-1} x_{i0k} = 1 \, , k \in Z_m$$

2. Setiap kota selain depot harus dikunjungi tepat satu kali. Kendala ini diekspresikan sebagai:

$$\sum_{j=0}^{n-1}\sum_{k=1}^{m} x_{ijk} = 1\,, i \neq j, i \in V \setminus \{0\}$$

$$\sum_{j=0}^{n-1}\sum_{k=1}^{m} x_{jik} = 1\,, i \neq j, i \in V \setminus \{0\}$$

3. Pada CTSP wilayah kerja terbagi menjadi dua, salah satunya yaitu wilayah pribadi yang hanya boleh dikunjungi oleh pekerja yang ditugaskan, sehingga pekerja k tidak boleh mengunjungi wilayah pribadi milik pekerja lain dari wilayah pribadinya, dan pekerja selain pekerja k tidak boleh mengunjungi wilayah pribadi pekerja k yang tersisa. Kendala ini diekspresikan sebagai:

$$\sum_{i}\sum_{j} x_{ijk} = 0\,, i \in P_k, j \in V \setminus (U \cup P_k), k \in Z_m$$

$$\sum_{i}\sum_{j} x_{jik} = 0\,, i \in P_k, j \in V \setminus (U \cup P_k), k \in Z_m$$

4. Pekerja $l(l \neq k)$ tidak boleh berangkat dari ataupun kembali ke wilayah pribadi pekerja k. Kendala ini diekspresikan sebagai:

$$\sum_{i}\sum_{j} x_{ijl} = 0\,, i \neq j, k \neq l, i \in P_k, j \in V, l \in Z_m$$

$$\sum_{i}\sum_{j} x_{jil} = 0\,, i \neq j, k \neq l, i \in P_k, j \in V, l \in Z_m$$

5. Wilayah umum merupakan wilayah yang dapat dikunjungi oleh setiap pekerja. Seorang pekerja dapat mengunjungi wilayah umum dan sepasang akses untuk masuk dan keluar jika dibutuhkan. Kendala ini diekspresikan sebagai:

$$\sum_{l} x_{jlk} = \sum_{i} x_{ijk}\,, i \neq j \neq l, j \in U, l \in P_k \cup U$$

6. Rute dari setiap pekerja harus terhubung dan tidak boleh ada sub-rute yang tidak terhubung pada kota-kota $V \setminus \{0\}$ sehingga $(u_{ik} - u_{jk} + n) \times x_{ijk} \leq n - 1, j \neq i, i, j \in V \setminus \{0\}, k \in Z_m$ dengan u_{ik} adalah banyaknya kota yang dikunjungi pekerja k sebelum mencapai kota i dan u_{jk} adalah banyaknya kota yang dilewati pekerja k sebelum mencapai kota j

Selengkapnya model CTSP dituliskan sebagai model optimisasi sebagai berikut:

Meminimumkan:

$$f = \sum_{k=1}^{m} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} C_{ij} x_{ijk} \quad (3.1)$$

terhadap

$$\sum_{j=1}^{n-1} x_{0jk} = 1 \,, k \in Z_m \quad (3.2)$$

$$\sum_{i=1}^{n-1} x_{i0k} = 1 \,, k \in Z_m \quad (3.3)$$

$$\sum_{j=0}^{n-1} \sum_{k=1}^{m} x_{ijk} = 1 \,, i \neq j, i \in V \setminus \{0\} \quad (3.4)$$

$$\sum_{j=0}^{n-1} \sum_{k=1}^{m} x_{jik} = 1 \,, i \neq j, i \in V \setminus \{0\} \quad (3.5)$$

$$\sum_{i} \sum_{j} x_{ijk} = 0 \,, i \in P_k, j \in V \setminus (U \cup P_k), k \in Z_m \quad (3.6)$$

$$\sum_{i} \sum_{j} x_{jik} = 0 \,, i \in P_k, j \in V \setminus (U \cup P_k), k \in Z_m \quad (3.7)$$

$$\sum_{i} \sum_{j} x_{ijl} = 0 \,, i \neq j, k \neq l, i \in P_k, j \in V, l \in Z_m \quad (3.8)$$

$$\sum_{i} \sum_{j} x_{jil} = 0 \,, i \neq j, k \neq l, i \in P_k, j \in V, l \in Z_m \quad (3.9)$$

$$\sum_{l} x_{jlk} = \sum_{i} x_{ijk} \,, i \neq j \neq l, j \in U, l \in P_k \cup U \quad (3.10)$$

$$\left(u_{ik} - u_{jk} + n\right) \times x_{ijk} \leq n - 1, j \neq i, i, j \in V \setminus \{0\}, k \in Z_m \quad (3.11)$$

3.3 Algoritma Genetik *Hill-Climbing* pada CTSP

Colored Traveling Salesman Problem merupakan salah satu permasalahan optimasi kombinatorial yang termasuk sebagai *NP-Hard Problem*. Permasalahan ini memiliki banyak kemungkinan solusi dan membutuhkan waktu yang lama dalam penyelesaiannya agar menemukan solusi yang optimal, sehingga dibutuhkan metode

penyelesaian alternatif agar permasalahan ini dapat diselesaikan dengan cepat.

Pada penelitian ini, model CTSP pada persamaan (3.1) hingga (3.11) akan diselesaikan dengan menggunakan Algoritma Genetika *Hill-Climbing*. Algoritma Genetika *Hill-Climbing* adalah sebuah metode yang telah dikembangkan dari Algoritma Genetika Klasik dengan menyisipkan teknik *Hill-Climbing* dengan tujuan lebih mengoptimalkan solusi.

Pada Algoritma Genetika *Hill-Climbing* solusi akan direpresentasikan sebagai kumpulan dari kromosom yaitu individu dengan kota akan direpresentasikan sebagai gen. Pada tahap awal akan dibangkitkan individu-individu secara acak dalam sebuah populasi. Selanjutnya himpunan-himpunan solusi akan dihitung nilai *fitness*-nya, yang kemudian dipilih individu dengan nilai *fitness* terbaik. Pada kasus CTSP, individu yang terbaik adalah individu dengan nilai *fitness* paling tinggi. Individu yang telah terpilih akan dioptimalkan dengan menggunakan *Hill-Climbing* sebelum masuk ke dalam tahap *crossover* dan menghasilkan anak sebagai solusi baru yang membawa sifat dari induknya. Dalam beberapa generasi, akan bermunculan individu-individu terbaik, semakin banyak perkawinan silang akan semakin banyak kemungkinan solusi terbaik yang diperoleh. Setelah melalui tahapan-tahapan dalam Algoritma Genetika *Hill-Climbing*, diharapkan akan ditemukan solusi dengan nilai *fitness* yang semakin tinggi dibandingkan dengan generasi sebelumnya.

3.3.1 Representasi Kromosom

Pada permasalahan CTSP, satu individu merepresentasikan satu solusi. Dalam individu terdapat beberapa kromosom dimana kromosom merepresentasikan rute yang dilalui oleh pekerja k. Permasalahan ini direpresentasikan menggunakan *Permutation Encoding*.

Sebagai contoh, misalkan terdapat 28 kota dimana 7 kota berada di masing-masing 3 wilayah pribadi dan 6 kota berada di wilayah umum dan 1 kota sebagai depot dengan rincian sebagai berikut:
Depot = {0}
Wilayah Umum = {1,2,3,4,5,6}
Wilayah Pribadi 1 = {7,8,9,10,11,12,13}
Wilayah Pribadi 2 = {14,15,16,17,18,19,20}
Wilayah Pribadi 3 = {21,22,23,24,25,26,27}
Setiap pekerja k akan mengunjungi kota dari setiap rute sesuai dengan wilayah pribadi masing-masing dan wilayah umum yang ditugaskan dengan berangkat dari depot dan kembali ke depot. Rute setiap pekerja direpresentasikan sebagai kromosom dengan kota yang dikunjungi

sebagai gen. Berdasarkan asumsi banyaknya kota yang dikunjungi oleh setiap pekerja haruslah sama, sehingga pada satu individu untuk setiap solusi terdapat 3 kromosom yang memuat 9 gen dimana 7 gen merupakan wilayah pribadi dan 2 gen wilayah umum.

Pada model CTSP terdapat beberapa kendala yang harus dipenuhi dan akan diterapkan pada Algoritma Genetika. Persamaan (3.1) akan digunakan sebagai fungsi untuk menghitung nilai *fitness* tiap individu dengan diawali oleh depot yang direpresentasikan oleh bilangan 0 sesuai dengan persamaan (3.2) dan (3.3). Kendala pada persamaan (3.6) sampai (3.9) akan dinyatakan dengan mengisi kromosom dengan gen atau kota yang telah ditugaskan pada pekerja tersebut. Selanjutnya karena jumlah gen pada tiap kromosom harus sama, maka wilayah pribadi akan dibagi secara rata dan acak pada masing-masing kromosom yang merepresentasikan kendala pada persamaan (3.10). Berdasarkan persamaan (3.4) dan (3.5), setiap kota harus dikunjungi tepat satu kali sehingga tidak boleh ada gen dengan bilangan yang sama pada setiap individu. Representasi dapat dilihat pada Gambar 3.1.

Kromosom 1	8	2	7	3	10	11	9	12	13
Kromosom 2	1	14	15	19	17	20	16	18	5
Kromosom 3	24	27	23	25	22	21	26	4	6

Gambar 3. 1. Representasi Kromosom

Pada kromosom Gambar 3.1 dapat diinterpretasikan bahwa pekerja 1 memiliki rute : 0-8-2-7-3-10-11-9-12-13-0, pekerja 2 memiliki rute : 0-1-14-15-19-17-20-16-18-0, dan pekerja 3 memiliki rute : 0-1-14-15-19-17-20-16-18-5-0.

3.3.2 Pembangkitan Populasi Awal

Populasi awal dibangkitkan secara acak sesuai dengan jumlah populasi yang telah ditentukan. Populasi berisi beberapa individu yang telah didefinisikan sesuai dengan representasi kromosom. Selanjutnya ditentukan parameter dari Algoritma Genetika yaitu generasi, probabilitas *crossover* dan probabilitas mutasi. Generasi menentukan seberapa banyak pengulangan yang akan dilakukan pada tahap dimulai dari menghitung nilai *fitness* hingga mutasi.

3.3.3 Nilai *Fitness*

Nilai *fitness* mengukur seberapa optimal individu yang dibangkitkan. Pada permasalahan CTSP semakin tinggi nilai *fitness* maka semakin baik solusinya. Fungsi *fitness* yang digunakan yaitu

$$F(x) = \frac{1}{f(x)}$$

dimana F(x) adalah fungsi *fitness* individu x dan f(x) adalah fungsi tujuan dari CTSP.

Langkah untuk menghitung fungsi tujuan dari individu yaitu dengan menjumlahkan total jarak dari setiap kromosom. Misalkan pada Kromosom 1, akan dihitung jarak dari 0 ke 8 kemudian ditambahkan dengan jarak dari 8 ke 2, 2 ke 7, 7 ke 3, 3 ke 10, 10 ke 11, 11 ke 9, 9 ke 12, 12 ke 13 dan 13 ke 0, kemudian lakukan langkah yang sama pada Kromosom 2 dan Kromosom 3. Fungsi tujuan dari individu pada Gambar 3.1 merupakan jumlah dari total jarak Kromosom 1, Kromosom 2 dan Kromosom 3.

3.3.4 Algoritma *Hill-Climbing*

Setelah populasi dibangkitkan dan masing-masing individu dihitung nilai *fitness*-nya, tahap pertama yang dilakukan yaitu Algoritma *Hill-Climbing*. Setiap individu akan dibangkitkan populasi yang merupakan kombinasi lain dari individu itu sendiri kemudian dihitung nilai *fitness*-nya dan dipilih individu dengan nilai *fitness* yang lebih tinggi. Misalkan jumlah individu dalam populasi adalah p, i menentukan pekerja dan a menentukan individu. Langkah-langkah Algoritma *Hill-Climbing* pada Algoritma Genetika dengan permasalahan CTSP ini adalah sebagai berikut:

1. Tetapkan $a = 1$, dan $i = 1$.
2. Jika $i < m$ maka lakukan langkah 3. Jika $i = m$ maka $a = a + 1$ lalu kembali ke langkah 3. Jika $a = p$ maka berhenti.
3. Pilih 2 gen (kota) yang telah ditugaskan pada pekerja i, dari kromosom kota pada individu a. Tukar kedua gen tersebut kemudian diperoleh individu a'.
4. Periksa apakah nilai *fitness* dari a' lebih besar dari a. Jika iya maka $a = a'$, sebaliknya maka $a = a$.
5. Tetapkan $i = i + 1$ dan kembali ke langkah 2.

Langkah selanjutnya yaitu hitung nilai *fitness* dari masing-masing individu dan pilih individu dengan nilai *fitness* paling tinggi, apabila individu yang terpilih tersebut telah memenuhi syarat dan kendala maka individu tersebut merupakan solusi optimal.

3.3.5 Seleksi

Tahapan seleksi akan dilakukan jika individu yang terpilih dari hasil Algoritma *Hill-Climbing* tidak memenuhi syarat dan kendala yang telah ditentukan. Seleksi merupakan proses untuk memilih individu yang akan menjadi induk pada tahap *crossover*. Pada penelitian ini proses

seleksi yang digunakan yaitu *Roulette Wheel*. Langkah pertama yaitu menghitung probabilitas dari masing-masing individu dengan rumus:

$$\text{Probabilitas Individu } i = \frac{\text{Nilai } fitness \text{ individu } i}{\text{Total nilai } fitness \text{ populasi}}$$

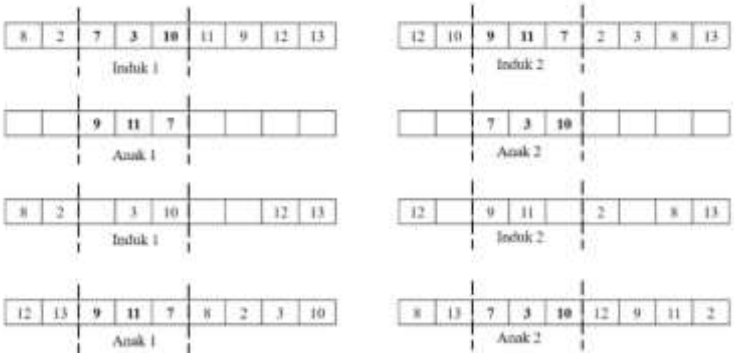
Selanjutnya setelah masing-masing individu dihitung probabilitasnya, setiap individu akan diberi jatah bilangan dari 1 sampai 1000 sesuai dengan nilai probabilitasnya. Kemudian dibangkitkan nilai random dari 1 sampai 1000 sebanyak populasi awal dan diperoleh populasi dengan individu-individu baru.

3.3.6 *Crossover*

Crossover merupakan tahapan setelah Seleksi dimana pada tahap ini dua individu induk akan dikawinkan dan memperoleh dua anak sebagai individu baru. Individu yang digunakan sebagai induk dipilih dengan menggunakan Probabilitas *Crossover* (ρc) yang telah ditentukan. Kemudian pada masing-masing individu akan dibangkitkan bilangan acak P dari 0 sampai 1, jika nilai bilangan acak P kurang dari ρc maka individu tersebut terpilih untuk melakukan persilangan. Pada penelitian ini proses *Crossover* yang akan digunakan yaitu *Order Crossover (OX)*.

Langkah-langkah dalam proses *Order Crossover* adalah sebagai berikut:

1. Tentukan dua individu yang akan menjadi induk.
2. Tentukan kromosom yang akan di *crossover* dari individu lalu tentukan titik *crossover* dengan membangkitkan dua bilangan bulat acak dari 1 sampai banyaknya gen. Misal muncul angka 3 dan 5, maka titik *crossover* merupakan gen ke 3 hingga gen ke 5.
3. Selanjutnya buat kromosom kosong sebagai anak 1 dan anak 2, kemudian tempatkan gen ke 3 hingga gen ke 5 induk 1 pada kromosom anak 2 dan sebaliknya dengan posisi yang sama.
4. Kemudian posisi kosong pada anak 1 akan diisi oleh gen yang belum termuat pada anak 1 dari induk 1 secara terurut dari gen setelah titik *crossover* dan juga pada anak 2.



Gambar 3. 2. Langkah-langkah *Crossover*

3.3.7 Mutasi

Pada Algoritma Genetika seluruh individu baru hasil dari *crossover* akan mengalami mutasi. Gen yang akan dimutasi dipilih menggunakan Probabilitas Mutasi (ρm). Setiap gen akan dibangkitkan nilai acak p dari 0 sampai 1, jika nilai bilangan acak p kurang dari ρm maka akan dilakukan mutasi pada gen tersebut. Selanjutnya, proses mutasi yang digunakan dalam penelitian ini yaitu *Swapping Mutation* sehingga gen yang akan dimutasi ditukar dengan gen yang dipilih secara acak. Proses ini dilakukan pada setiap gen dari setiap kromosom pada individu hasil dari *crossover*.

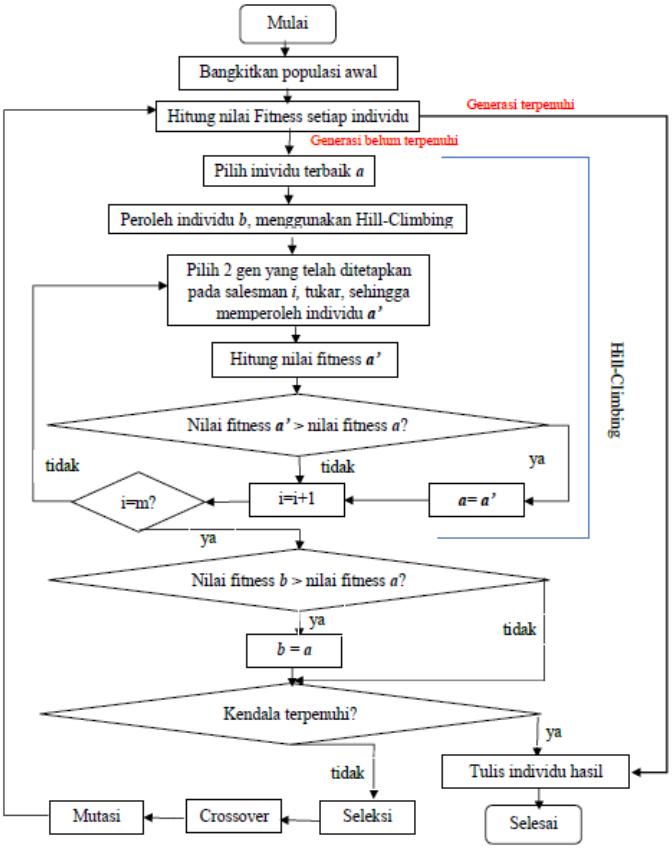
Misal $\rho m = 0{,}5$ kemudian bangkitkan bilangan acak r dari 1 hingga jumlah gen dalam satu kromosom pada setiap gen. Misalkan bilangan acak untuk gen ke 1 yaitu p=0,4, karena $p < \rho m$ maka akan dibangkitkan bilangan acak r=3 sehingga akan dilakukan penukaran gen ke 1 dengan gen ke 3.

Gambar 3. 3. Ilustasi Mutasi

3.3.8 Evaluasi

Pada tahap evaluasi akan diperiksa parameter generasi Algoritma Genetika, apabila Generasi belum terpenuhi maka lakukan pengulangan dari menghitung nilai *fitness*. Apabila telah dipenuhi maka tahap selanjutnya yaitu menghitung nilai *fitness* tiap individu akhir. Kemudian individu dengan nilai *fitness* paling tinggi akan terpilih sebagai solusi optimal dari Algortima Genetika *Hill-Climbing*.

Flowchart Algoritma Genetika *Hill-Climbing* dapat digambarkan sebagai berikut:



Gambar 3. 4. *Flowchart* Algoritma Genetika *Hill-Climbing*