PERBANDINGAN ALGORITMA *BRANCH AND BOUND* DAN ALGORITMA GENETIKA UNTUK MENGATASI *TRAVELLING SALESMAN PROBLEM* (TSP) MENGGUNAKAN *SOFTWARE* MATLAB (Studi Kasus PT. JNE Semarang)

Skripsi
disajikan sebagai salah satu syarat
untuk memperoleh gelar Sarjana Sains
Program Studi Matematika

oleh
Ari Yulianto Nugroho
4111411034

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI SEMARANG

2015

PERNYATAAN KEASLIAN TULISAN

Saya menyatakan bahwa skripsi ini bebas plagiat, dan apabila di kemudian hari terbukti terdapat plagiat dalam skripsi ini, maka saya bersedia menerima sanksi sesuai ketentuan peraturan perundang-undangan.

Semarang,25 September 2015

Ari Yulianto Nugroho

4111411034

PENGESAHAN

Skripsi yang berjudul

PERBANDINGAN ALGORITMA *BRANCH AND BOUND* DAN ALGORITMA GENETIKA UNTUK MENGATASI *TRAVELLING SALESMAN PROBLEM* (TSP) MENGGUNAKAN *SOFTWARE* MATLAB (Studi Kasus PT. JNE Semarang)

disusun oleh

Ari Yulianto Nurgoho

4111411034

telah dipertahankan di hadapan sidang Panitia Ujian Skripsi FMIPA UNNES pada tanggal 25 September 2015

Panitia:

Ketua	Sekretaris
Prof. Dr. Wiyanto, M.Si.	Drs. Arief Agoestanto,M.Si.
196310121988031001	196807221993031005

Penguji Utama

Endang Sugiharti, S.Si., M.Kom.
197401071999032001

Anggota Penguji/ Pembimbing Utama	Anggota Penguji/ Pembimbing Pendamping
Drs. Amin Suyitno, M.Pd.	Riza Arifudin,S.Pd.,M.Cs.
195206041976121001	198005252005011001

MOTTO DAN PERSEMBAHAN

Sebaik-baik manusia adalah yang paling bermanfaat bagi orang lain.

~HR Qudha'i dari Jabir Ra

Hai orang-orang yang beriman, mintalah pertolongan (kepada Allah) dengan sabar dan shalat, sesungguhnya Allah menyertai orang-orang yang sabar.

~Al-Baqarah: 153

Sesungguhnya sesudah kesulitan itu ada kemudahan, maka apabila kamu telah selesai (dari sesuatu urusan), kerjakanlah dengan sungguh-sungguh (urusan yang lain). Dan hanya kepada Allah-lah hendaknya kamu berharap.

~ Al-Insyirah: 6-8

Kupersembahkan skripsi ini untuk:

Bapak & Ibu

yang selalu menyayangi, mendukung dan tak pernah lelah mendoakan.

Kakak tersayang

dengan semua kedewasaan yang selalu menginspirasi.

- Untuk seluruh keluarga besarku yang selalu mendoakan.
- Sahabat-sahabat matematika angkatan 2011 yang telah menjadi penyemangat.
- Semua pihak yang tidak dapat disebutkan satu persatu yang telah membantu hingga terselesaikannya penulisan skripsi ini.

PRAKATA

Assalamu'alaikum, Wr. Wb.

Puji syukur penulis panjatkan ke hadirat Allah SWT karena berkat rahmat dan karunia-Nya, penulis dapat menyelesaikan penulisan skripsi ini dengan baik dan lancar. Skripsi yang berjudul "Perbandingan Algoritma *Branch and Bound* dan Algoritma Genetika untuk Mengatasi *Travelling Salesman Problem* (TSP) menggunakan *Software* Matlab (studi kasus PT. JNE Semarang)" disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata Satu (S1) pada Program Studi Matematika Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.

Dalam mengerjakan dan menyusun skripsi ini, penulis telah banyak mendapat bantuan, bimbingan, dorongan, dan petunjuk yang sangat bermanfaat dari berbagai pihak yang sangat mendukung. Oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih dengan tulus kepada:

1. Prof. Dr. Fathur Rokhman, M. Hum., Rektor Universitas Negeri Semarang.
2. Prof. Dr. Wiyanto, M.Si., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Drs. Arief Agoestanto, M.Si., Ketua Jurusan Matematika FMIPA UNNES.
4. Dra. Kristina Wijayanti, M.S., Ketua Program Studi Matematika UNNES.

5. Drs. Amin Suyitno, M.Pd., dan Riza Arifudin, S.Pd., M.Cs., selaku dosen pembimbing yang telah memberikan bimbingan, arahan, dan saran kepada penulis selama penyusunan skripsi.
6. Putriaji Hendikawati, S.Si., M.Pd., M.Sc., Dosen wali yang telah memberikan arahan dan motivasi sepanjang perjalanan penulis menimba ilmu di Universitas Negeri Semarang
7. Seluruh pihak yang telah membantu baik langsung maupun tidak langsung sehingga skripsi ini dapat disusun.

Semoga skripsi ini dapat membawa manfaat bagi penulis sendiri khususnya dan bagi para pembaca pada umumnya.

Wassalamu'alaikum. Wr. Wb.

Semarang,25 September 2015

Penulis

ABSTRAK

Nugroho, A. Y. 2015.*Perbandingan Algoritma Branch and Bound dan Algoritma Genetika untuk Mengatasi Travelling Salesman Problem (TSP) Menggunakan Software Matlab (Studi Kasus PT. Jalur Nugraha Ekakurir (JNE) Semarang).* Skripsi, Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Utama Drs. Amin Suyitno, M.Pd dan Pembimbing Pendamping Riza Arifudin, S.Pd., M.Cs.

Kata kunci: Graf, Algoritma *Branch and Bound,* Algoritma genetika, *Travelling Salesman Problem, software* Matlab.

Travelling Salesman Problem (TSP) adalah *problem* pencarian rute optimal bagi seseorang *salesman* yang berkeliling mengunjungi n kota dengan setiap kota dikunjungi satu kali kecuali kota asal. Penelitian ini fokus pada masalah pendistribusian barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang dengan tujuan 9 alamat pengiriman di wilayah Semarang.

Pengukuran keefektifan hasil kerja sistem dilakukan dengan membandingkan hasil perhitungan algoritma *branch and bound* dengan hasil perhitungan algoritma genetika yang merupakan hasil modifikasi terbaik. 20 Modifikasi pada algoritma genetika dilakukan pada ukuran populasi, besar probabilitas *crossover*, besar probabilitas mutasi dan jumlah generasinya.

Penelitian ini menghasilkan simpulan yaitu penerapan algoritma *branch and bound* dan algoritma genetika guna menentukan sirkuit terpendek dalam pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) semarang dimulai dengan mencari jarak antar alamat dengan bantuan *Google Maps*, kemudian dilanjutkan dengan pembangunan sistem TSP yang dilengkapi dengan koding pada *software* Matlab R2009a. Setelah sistem TSP berhasil dibuat selanjutnya inputkan data alamat yang telah disimpan dalam *database*, *inputkan* pula variabel-variabel masukan seperti ukuran populasi, probabilitas *crossover*, probabilitas mutasi dan generasi. Selanjutnya dilakukan pengujian sistem dengan melakukan modifikasi pada algoritma genetika. Hasil modifikasi pada algoritma genetika yang paling optimal digunakan untuk perbandingan dengan hasil perhitungan algoritma *branch and bound.* Selanjutnya akan didapatkan hasil perbandingan algoritma *branch and bound* dan algoritma genetika yang mempunyai panjang sirkuit terpendek. Berdasarkan solusi optimum yang diperoleh dengan menggunakan algoritma *branch and bound* dan dengan algoritma genetika diperoleh panjang sirkuit yang dihasilkan algoritma genetika lebih kecil dari panjang sirkuit yang dihasilkan algoritma *branch and bound.* Hal ini menunjukkan bahwa algoritma genetika lebih efektif dalam menentukan sirkuit terpendek untuk pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang.

DAFTAR ISI

Halaman

HALAMAN JUDUL i
HALAMAN PERNYATAAN KEASLIAN ii
HALAMAN PENGESAHAN iii
HALAMAN MOTTO DAN PERSEMBAHAN iv
PRAKATA v
ABSTRAK vii
DAFTAR ISI viii
DAFTAR TABEL xv
DAFTAR GAMBAR xvii
DAFTAR LAMPIRAN xxi
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Maslah 5
1.3 Pembatasan Masalah 5
1.4 Tujuan Penelitian 6
1.5 Manfaat Penelitian 6
1.6 Sistematika Penulisan 7
BAB II TINJAUAN PUSTAKA 9
2.1 Teori Graf 9
2.1.1 Pengertian Graf 9
2.1.2 Jenis-Jenis Graf 10

2.1.2.1 Graf Sederhana 10
2.1.2.2 Graf Tidak Sederhana 11
2.1.2.3 Graf Berarah dan Berbobot 11
2.1.2.4 Graf Tidak Berarah dan Berbobot 12
2.1.2.5 Graf Berarah dan Tidak Berbobot 12
2.1.2.6 Graf Tidak Berarah dan Tidak Berbobot 13
2.1.3 Terminologi Graf 13
2.1.3.1. Bertetangga 13
2.1.3.2. Bersisian 14
2.1.3.3. Graf kosong 15
2.1.3.4. Derajat 15
2.1.3.5. Jalan 15
2.1.3.6. Jejak 16
2.1.3.7. Lintasan 16
2.1.3.8. Jalur 17
2.1.3.9. Sirkuit 17
2.1.4 Graf Euler dan Graf Semi Euler 18
2.1.5 Lintasan dan Sirkuit hamilton 18
2.1.6 Representasi Graf 19
2.1.6.1 *Adjacercy Matrix* 19
2.1.6.2 *Adjancercy List* 20
2.1.7 Graf Terhubung 20
2.1.8 Pohon 21

2.1.8.1. Pohon Rentang 22
2.1.8.2. Pohon Rentang Minimum 22
2.1.8.3. Pohon Berakar 22
2.1.8.4. Pohon Rentang Status 23
2.1.9 Lintasan Terpendek 24
2.2 *Travelling Salesman Problem* 25
2.2.1 Sejarah TSP 26
2.2.2 Contoh Perkembangan Masalah yang Muncul 26
2.3 Algoritma *Branch and Bound* 27
2.4 Algoritma Genetika 44
2.4.1 Struktur Umum Algoritma Genetika 47
2.4.2 Komponen-Komponen Algoritma Genetika 49
2.4.2.1. Skema Pengkodean 51
2.4.2.2. Nilai *Fitness* 53
2.4.2.3. Seleksi Induk 54
2.4.2.4. Pindah Silang 57
2.4.2.5. Mutasi 62
2.4.2.6. Elitisme 64
2.4.2.7. Pergantian populasi 64
2.4.3. Penentuan Parameter Algoritma 65
2.5 *Travelling Salesman Problem* dalam Algoritma Genetika 66
2.6 *Matrix Laboratory* 68
2.6.1 Menjalankan Matlab 69

2.6.2 Menggunakan Variabel 69
2.6.3 Mengenal GUI 70
2.6.3.1 *Toolbar* GUI 70
2.6.3.2 Komponen-Komponen GUI 73
2.6.4 Merancang *Interface* 75
2.6.5 Menyimpan *Interface* 76
2.6.6 Menjalankan *Interface* 77
BAB III METODE PENELITIAN 78
3.1 Identifikasi Masalah 78
3.2 Rumusan Masalah 78
3.3 Pemecahan Masalah 79
3.3.1 Pengambilan Data 79
3.3.2 Pengolahan Data 81
3.4 Penarikan Kesimpulan 87
BAB IV HASIL DAN PEMBAHASAN 88
4.1 Hasil Penelitian 88
4.1.1 Penyelesaian Masalah dengan Algoritma *Branch and Bound* . 89
4.1.2 Penyelesaian Masalah dengan Algoritma Genetika 106
4.1.3 Implementasi Program 116
4.1.3.1.*Form* Halaman Depan 116
4.1.3.2.*Form* Halaman Utama 117
4.2 Pembahasan 119
4.2.1 Pengujian Sistem 119

4.2.1.1.Perhitungan Menggunakan Algoritma *Branch and Bound* 120
4.2.1.2.Perhitungan Menggunakan Algoritma Genetika..... 122
4.2.1.2.1. Perhitungan Menggunakan Modifikasi Ukuran Populasi 50, Probabilitas *Crossover* 0,6, Probabiitas Mutasi 0,1 dan Generasi 50 122
4.2.1.2.2. Perhitungan Menggunakan Modifikasi Ukuran Populasi 100, Probabilitas *Crossover* 0,6, Probabiitas Mutasi 0,1 dan Generasi 50 125
4.2.1.2.3. Perhitungan Menggunakan Modifikasi Ukuran Populasi 150, Probabilitas *Crossover* 0,7, Probabiitas Mutasi 0,1 dan Generasi 50 125
4.2.1.2.4. Perhitungan Menggunakan Modifikasi Ukuran Populasi 50, Probabilitas *Crossover* 0,7, Probabiitas Mutasi 0,1 dan Generasi 150 126
4.2.1.2.5. Perhitungan Menggunakan Modifikasi Ukuran Populasi 100, Probabilitas *Crossover* 0,6, Probabiitas Mutasi 0,1 dan Generasi 150 127
4.2.1.2.6. Perhitungan Menggunakan Modifikasi Ukuran Populasi 150, Probabilitas *Crossover* 0,6, Probabiitas Mutasi 0,1 dan Generasi 150 128

4.2.1.2.7. Perhitungan Menggunakan Modifikasi Ukuran Populasi 50, Probabilitas *Crossover* 0,7, Probabiitas Mutasi 0,1 dan Generasi 50 129
4.2.1.2.8. Perhitungan Menggunakan Modifikasi Ukuran Populasi 100, Probabilitas *Crossover* 0,7, Probabiitas Mutasi 0,1 dan Generasi 50 129
4.2.1.2.9. Perhitungan Menggunakan Modifikasi Ukuran Populasi 150, Probabilitas *Crossover* 0,6, Probabiitas Mutasi 0,1 dan Generasi 50 130
4.2.1.2.10. Perhitungan Menggunakan Modifikasi Ukuran Populasi 50, Probabilitas *Crossover* 0,6, Probabiitas Mutasi 0,1 dan Generasi 150 131
4.2.1.2.11. Perhitungan Menggunakan Modifikasi Ukuran Populasi 100, Probabilitas *Crossover* 0,7, Probabiitas Mutasi 0,1 dan Generasi 150 132
4.2.1.2.12. Perhitungan Menggunakan Modifikasi Ukuran Populasi 50, Probabilitas *Crossover* 0,7, Probabiitas Mutasi 0,1 dan Generasi 150 133
4.2.2 Analisis Hasil Kerja 134
BAB V PENUTUP..... 137
5.1 Kesimpulan 137
5.2 Saran..... 138
DAFTAR PUSTAKA 140

LAMPIRAN 143

DAFTAR TABEL

Tabel		Halaman
4.1	Nama Alamat dan Koordinat Lokasi	89
4.2	Data Jarak Antar Alamat Tujuan (dalam Satuan Meter)	89
4.3	Populasi Awal.....	107
4.4	Populasi Setelah Melalui Proses Pengurutan Sesuai Nilai *Fitness* Terbaik	109
4.5	Populasi Setelah Proses *Crossover*	114
4.6	Populasi Setelah Proses Mutasi	116
4.7	Hasil Perhitungan Menggunakan UkPop 50, Pc 0,6, Pm 0,1 dan Gen 50...	124
4.8	Hasil Perhitungan Menggunakan UkPop 100, Pc 0,6, Pm 0,1 dan Gen 50.	125
4.9	Hasil Perhitungan Menggunakan UkPop 150, Pc 0,7, Pm 0,1 dan Gen 50.	126
4.10	Hasil Perhitungan Menggunakan UkPop 50, Pc 0,7, Pm 0,1 dan Gen 150.	126
4.11	Hasil Perhitungan Menggunakan UkPop 100 Pc 0,6, Pm 0,1 dan Gen 150	127
4.12	Hasil Perhitungan Menggunakan UkPop 150 Pc 0,6, Pm 0,1 dan Gen 150	128
4.13	Hasil Perhitungan Menggunakan UkPop 50, Pc 0,7, Pm 0,1 dan Gen 50...	129
4.14	Hasil Perhitungan Menggunakan UkPop 100, Pc 0,7, Pm 0,1 dan Gen 50.	130
4.15	Hasil Perhitungan Menggunakan UkPop 150, Pc 0,6, Pm 0,1 dan Gen 50.	130
4.16	Hasil Perhitungan Menggunakan UkPop 50, Pc 0,6, Pm 0,1 dan Gen 150.	131
4.17	Hasil Perhitungan Menggunakan UkPop 100, Pc 0,7, Pm 0,1 dan Gen 150	132
4.18	Hasil Perhitungan Menggunakan UkPop 150, Pc 0,7, Pm 0,1 dan Gen 150	133
4.19	Hasil Panjang Sirkuit Terpendek pada 12 Modifikasi Algoritma Genetika	134

4.20 Hasil Perhitungan *Travelling Salesman Problem* dengan Algoritma *Branch and Bound* dan Algoritma Genetika..... 135

DAFTAR GAMBAR

Halaman

Gambar 2.1 Contoh Graf 10

Gambar 2.2 Graf Sederhana 10

Gambar 2.3 Graf Tak Sederhana 11

Gambar 2.4 Graf Berarah dan Berbobot 11

Gambar 2.5 Graf Tidak Berarah dan berbobot 12

Gambar 2.6 Graf Berarah dan Tidak Berbobot 12

Gambar 2.7 Graf Tidak Berarah dan Tidak Berbobot 13

Gambar 2.8 Graf Bertetangga 14

Gambar 2.9 Graf Bersisian 14

Gambar 2.10 Graf Kosong 15

Gambar 2.11 Jalan Pada Graf G $(V_3 - V_5 - V_4 - V_2 - V_1 - V_3)$ 16

Gambar 2.12 Jejak Pada Graf G$(V_3 - V_5 - V_2 - V_1 - V_5 - V_4)$ 16

Gambar 2.13 Lintasan Pada Graf G$(V_1 - V_3 - V_5 - V_2 - V_4)$ 17

Gambar 2.14 Sirkuit Pada Graf G$(V_1 - V_3 - V_5 - V_4 - V_2 - V_1)$ 17

Gambar 2.15 Euler Pada Graf 18

Gambar 2.16 Hamilton Pada Graf G 19

Gambar 2.17 Graf G dan Matrix Berhubungan langsungnya 20

Gambar 2.18 Graf Terhubung, Graf Tak Terhubung 21

Gambar 2.19 Contoh Graf Pada Pohon dan Bukan Pohon 21

Gambar 2.20 Contoh Pohon-Pohon Rentang pada Graf G 22

Gambar 2.21 T_1 Pohon Rentang Minimum Dari Graf G 22

Gambar 2.22 Macam-Macam Pohon Berakar 23
Gambar 2.23 Pohon Ruang Status 24
Gambar 2.24 Graf *Breadth first search* 28
Gambar 2.25 Graf yang Menggambarkan Posisi Titik-Titik Tujuan 31
Gambar 2.26 Pohon Ruang Status Level 0 32
Gambar 2.27 Pohon Ruang Status Level 1 36
Gambar 2.28 Pohon Ruang Status Level 2 38
Gambar 2.29 Pohon Ruang Status Level 3 41
Gambar 2.30 Pohon Ruang Status Level 4 42
Gambar 2.31 Pohon Ruang Status Level 5 43
Gambar 2.32 Ilustrasi Definisi Individu Dalam Algoritma Genetika 46
Gambar 2.33 Diagram Alir Algoritma Genetika 49
Gambar 2.34 Siklus Algoritma Genetika 50
Gambar 2.35 Skema Pengkodean *Binary Encoding* 51
Gambar 2.36 Skema Pengkodean *Discrete Decimal Ecoding* 52
Gambar 2.37 Skema Pengkodean *Real-Number Encoding* 52
Gambar 2.38 Contoh Penggunaan Metode *Roulette-Wheel* 55
Gambar 2.39 Ilustrasi Prinsip Kerja Metode *Roulette-Wheel* 56
Gambar 2.40 Contoh Proses Pindang Silang 57
Gambar 2.41 Gen Sebelum dan Setelah Mutasi dengan Pengkodean Pohon .. 63
Gambar 2.42 Pindah Silang Menggunakan *Order Crossover* 67
Gambar 2.43 Tampilan Lembar Kerja GUI 70
Gambar 2.44 Tampilan *Menu Editor* 71

Gambar 2.45 Tampilan *Align Object* 72

Gambar 2.46 Tampilan *Property Inspector* 72

Gambar 2.47 Komponen-Komponen GUI..... 73

Gambar 3.1 Halaman Depan *Google Maps* 80

Gambar 3.2 Pengolahan Data..... 81

Gambar 3.3 Diagram Alir Algoritma Branch and Bound..... 82

Gambar 3.4 Diagram Alir Algoritma Genetika..... 84

Gambar 4.1 Graf yang Menggambarkan Titik-Titik Tujuan 90

Gambar 4.2 Pohon Ruang Status Level 0 92

Gambar 4.3 Pohon Ruang Status Level 1 99

Gambar 4.4 Pohon Ruang Status Level 2 104

Gambar 4.5 Pohon Ruang Status Level 9 105

Gambar 4.6 Tampilan *Form* Halaman Depan..... 117

Gambar 4.7 Tampilan *Form* Halaman Utama (TSP)..... 117

Gambar 4.8 Tampilan Hasil Uji 118

Gambar 4.9 Kotak Dialog Keluar Aplikasi..... 119

Gambar 4.10 Tampilan TSP Setelah Memasukkan Koordinat Alamat 120

Gambar 4.11 Tampilan TSP Beserta Grafik Sirkuit Koordinat Alamat Tujuan pada Algoritma *Branch and Bound*..... 121

Gambar 4.12 Tampilan TSP Setelah Memasukkan Koordinat Alamat, Ukuran Populasi, Probabilitas *Crossover*, Probabilitas mutasi dan Generasi..... 122

Gambar 4.13 Tampilan TSP Besertas Grafik Sirkuit Koordinat Alamat..... 123

Gambar 4.14 Hasil Uji pada Ukuran Populasi 50, Probabilitas *Crossover* 0,6, Probabilitas Mutasi 0,1 dan Generasi 50 123

Gambar 4.15 Tampilan Data yang Telah Disimpan pada *Ms.Excel* 124

DAFTAR LAMPIRAN

Lampiran		Halaman
1	*Source Cod*e Halaman Depan.....	143
2	*Source Cod*e Halaman Utama (TSP).....	146
3	*Source Cod*e Halaman Hasil Uji.....	159

BAB 1

PENDAHULUAN

1.1. Latar Belakang

Matematika merupakan pengetahuan dasar dalam proses berkembangnya ilmu pengetahuan dan teknologi. Hampir setiap kegiatan membutuhkan peranan matematika. Tidak hanya itu, perkembangan teknologi yang kian pesat juga tidak lepas dari matematika. Dengan kata lain matematika merupakan pengetahuan dan teknologi baik dalam unsur kajian umum, ilmu murni, maupun terapannya.

Pencarian rute terpendek merupakan suatu masalah yang paling banyak dibahas dan dipelajari sejak akhir tahun 1950. Pencarian rute terpendek ini telah diterapkan di berbagai bidang untuk mengoptimasi kinerja suatu sistem, baik untuk meminimalkan biaya atau mempercepat jalannya suatu proses (Purwananto, 2005:1). Masalah rute terpendek secara umum dijelaskan menggunakan konsep graf, dapat berupa graf berarah atau graf tidak berarah. Sisi dalam sebuah graf tidak berarah dapat dianggap memungkinkan perjalanan di kedua arah. Sebaliknya, sisi dalam graf berarah hanya dapat digunakan untuk satu arah perjalanan.

Graf yang digunakan untuk menyelesaikan masalah rute terpendek antara dua atau beberapa titik yang berhubungan adalah graf berbobot (*weighted graph*), yaitu graf yang memiliki bobot atau nilai pada setiap sisinya. Bobot pada sisi graf

dapat menyatakan jarak antara dua alamat, waktu pengiriminan, ongkos pembangunan dan sebagainya. Asumsi yang digunakan di sini adalah bahwa semua bobot bernilai positif. Kata “terpendek” bukan berarti memiliki panjang minimum, akan tetapi kata “terpendek” disini memiliki pengertian meminimalisasi nilai graf berbobot dari alamat asal ke alamat tujuan dan kembali lagi ke alamat asal.

Kesulitan menentukan rute terpendek timbul karena terdapat banyak jalur yang ada dari suatu daerah ke daerah lain sehingga memungkinkan memilih jalur alternatif apabila terdapat suatu hambatan pada jalur terpendek utama. Kebutuhan untuk menemukan rute terpendek dan waktu tempuh tercepat tentunya juga diperhitungkan untuk menghindari kerugian seperti contoh bagi sebuah industri. Kebutuhan untuk segera sampai tempat tujuan tepat waktu bahkan diharapkan bisa lebih cepat sangatlah dibutuhkan mengingat persaingan industri yang mementingkan kepuasan pelanggan dan menghindari kerugian karena kerusakan barang.

Proses pendistribusian barang adalah kegiatan yang tidak pernah lepas dari kehidupan. Jarak yang jauh serta penyebaran masyarakat yang meluas menjadi salah satu alasan bagi masyarakat untuk menggunakan jasa pengiriman barang dari pada mengantar sendiri barang yang akan dikirimkan. Masalah pengiriman barang menjadi poin terpenting bagi perusahaan penyedia jasa pengiriman barang. Hal ini sangat memerlukan pertimbangan dan perhitungan yang tepat karena berkaitan dengan biaya transportasi yang harus dikeluarkan dalam proses pendistribusian (Sari, Dwijanto & Sugiharti, 2013:2).

PT. Jalur Nugraha Ekakurir (JNE) merupakan salah satu perusahaan yang bergerak dalam bidang jasa pengiriman barang di Indonesia. PT. Jalur Nugraha Ekakurir (JNE) sendiri memiliki cabang di setiap alamat di seluruh Indonesia. Dalam mengirim barang dari pusat ke pelanggan di berbagai tempat dan di banyak alamat, perlu adanya suatu sistem yang mampu meminimalisasi biaya pengiriman agar didapatkan keuntungan yang maksimal. Permasalahan seperti ini merupakan masalah model jaringan yang biasa disebut *Travelling Salesmam Problem.*

Travelling Salesman Problem (TSP) merupakan salah satu masalah optimalisasi. TSP adalah suatu permasalahan untuk menemukan siklus Hamilton yang memiliki total bobot sisi minimum. TSP bertujuan mencari rute dari alamat asal ke alamat yang dituju dengan syarat setiap alamat hanya dapat dikunjungi satu kali kecuali alamat awal. Banyak algoritma yang diterapkan pada permasalahan TSP di antaranya adalah *nearest neighbor heuristic, cheapest insertion heuristic, two way exchange improvement heuristic, nearest insertion heuristic, genetic, ant colony optimation, dan branch and bound method*.

Algoritma *branch and bound* pertama kali dikenalkan oleh A. H Land dan A. G Doig pada tahun 1960. Algoritma *Branch and Bound* adalah suatu prosedur yang paling umum untuk mencari solusi optimal pada masalah optimal kombinatorial seperti masalah penjadwalan. Dalam algoritma *Branch and Bound* terdapat tiga bagian utama, yaitu : ekspresi batas bawah (*lower bound*), strategi pencarian, dan pencabangan (Sutanto, 2011:2). Sedangkan algoritma Genetika pertama kali diperkenalkan oleh John Holland (1975) dari Universitas Michigan. John Holland mengatakan bahwa setiap masalah yang berbentuk adaptasi (alami

maupun buatan) dapat diformulasikan ke dalam terminologi genetika. Goldberg mendefinisikan algoritma Genetika ini sebagai suatu pencarian algoritma berdasarkan pada mekanisme seleksi alam dan genetika alam.

Software Matlab merupakan bahasa pemrograman yang hadir dengan fungsi dan karakteristik yang berbeda dengan bahasa pemrograman lain yang sudah ada. *Software* Matlab merupakan bahasa pemrograman level tinggi yang dikhususkan untuk kebutuhan komputasi teknis, visualisasi dan pemrograman seperti komputasi matematik, analisis data, pengembangan algoritma, simulasi dan pemodelan dan grafik-grafik perhitungan.

Software Matlab hadir dengan membawa warna yang berbeda. Hal ini karena *Software* Matlab membawa keistimewaan dalam fungsi-fungsi matematika, fisika, statistik, dan visualisasi. *Software* Matlab dikembangkan oleh MathWorks, yang pada awalnya dibuat untuk memberikan kemudahan mengakses data matrik pada proyek Linpack dan Eispack. Saat ini *Software* Matlab memiliki ratusan fungsi yang dapat digunakan sebagai *problem solver* mulai dari masalah yang simpel sampai masalah-masalah yang kompleks dari berbagai disiplin ilmu (Firmansyah, 2007).

Berdasarkan latar belakang masalah tersebut peneliti akan melakukan penelitian yang terformulasikan dalam skripsi yang berjudul "**PERBANDINGAN ALGORITMA *BRANCH AND BOUND* DAN ALGORITMA GENETIKA UNTUK MENGATASI *TRAVELLING SALESMAN PROBLEM* (TSP) MENGGUNAKAN *SOFTWARE* MATLAB (Studi Kasus PT. JNE Semarang)**"

1.2. Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan di atas, maka rumusan masalah yang akan dikaji dalam penelitian ini adalah sebagai berikut.

1. Bagaimana mengimplementasikan Algoritma *Branch and Bound* dan Algoritma Genetika guna menentukan sirkuit terpendek dalam pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang?
2. Algoritma manakah antara Algoritma *Branch and Bound* dan Algoritma Genetika yang memiliki tingkat keefektifan terbaik dalam menentukan sirkuit terpendek untuk pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang?

1.3. Pembatasan Masalah

Agar dalam pembahasan skripsi ini tidak terlalu meluas, maka peneliti mencantumkan pembatasan masalah sebagai berikut.

1. Penelitian hanya dilakukan pada PT. Jalur Nugraha Ekakurir (JNE) yang terletak di Jalan Kyai Saleh No. 10.
2. Permasalahan diasumsikan sebagai sebuah *Travelling Salesman Problem simetris*, dengan jarak dari alamat 1 ke alamat 2 sama dengan jarak dari alamat 2 ke alamat 1.
3. Perhitungan jarak antara lokasi dalam penelitian dilakukan dengan internet.
4. Penelitian menggunakan jarak yang sebenarnya dan digambarkan dalam bentuk graf.

5. *Software* yang digunakan dalam penelitian ini adalah *software* Matlab R2009a.

1.4. Tujuan Penelitian

Berdasarkan rumusan masalah yang telah diuraikan, maka tujuan dilakukannya penelitian ini adalah sebagai berikut.

1. Menentukan sirkuit terpendek untuk pengiriman barang dengan menggunakan Algoritma *Branch and Bound* dan Algoritma Genetika di PT. Jalur Nugraha Ekakurir Semarang.
2. Menentukan algoritma yang memiliki tingkat keefektifan terbaik untuk menyelesaikan sirkuit terpendek untuk pengiriman barang di PT. Jalur Nugraha Ekakurir Semarang.

1.5. Manfaat Penelitian

Manfaat dari penelitian ini adalah sebagai berikut.

1) Bagi Penulis

   Mengetahui dan memahami aplikasi teori graf terutama Algoritma *Branch and Bound* dan Algoritma Genetika terhadap pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang.

2) Bagi Universitas

   Dari hasil penelitian ini dapat menjadi referensi yang berkaitan dengan teori graf dalam menyelesaikan masalah menghitung masalah sirkuit terpendek

3) Bagi Masyarakat Umum

Memberikan wawasan dan pengetahuan tentang pendistribusian dan transportasi suatu jaringan *Travelling Salesman Problem*.

1.6. Sistematika Penulisan

Secara garis besar penulisan skripsi ini terdiri atas 3 bagian, yaitu bagian awal, bagian pokok dan bagian akhir yang masing-masing diuraikan sebagai berikut.

1. Bagian awal

   Dalam penulisan skripsi ini, bagian awal berisi halaman judul, halaman pengesahan, motto dan persembahan, kata pengantar, abstrak, daftar isi, daftar gambar, daftar tabel dan daftar lampiran.

2. Bagian pokok

   Bagian pokok dari penulisan skripsi ini adalah isi skripsi yang terdiri atas 5 bab, yaitu:

   BAB I Pendahuluan, berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, serta sistematika penulisan skripsi.

   BAB II Tinjauan Pustaka, menguraikan tentang definisi maupun pemikiran-pemikiran yang dijadikan kerangka teoritis yang mendasari pemecahan dari permasalahan pada penelitian. Bagian ini dibagi menjadi beberapa sub bab meliputi Teori graf, Lintasan terpendek, *Travelling Salesman Probrem* (TSP), Algoritma *Branch and Bound*, Algoritma Genetika dan *Matrix Laboratory*.

BAB III Metode Penelitian, menyajikan gagasan pokok yang terdiri dari tahap permasalahan, investigasi awal, persiapan penelitian, penyelesaian, tahap pelaporan hasil, dan penarikan kesimpulan.

BAB IV Hasil dan Pembahasan, berisi hasil dan pembahasan dari permasalahan yang dikemukakan. Bab ini dibagi menjadi dua sub bab, yaitu hasil penelitian dan pembahasan. Hasil penelitian berisi hasil perhitungan dan analisis data yang diperoleh dari studi pustaka maupun pemecahan kasus penentuan sirkuit terpendek dari jaringan TSP pada pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang menggunakan *software* Matlab.

BAB V Penutup, berisi simpulan dan saran-saran peneliti.

3. Bagian Akhir

Bagian akhir skripsi berisi daftar pustaka sebagai acuan penulisan dan lampiran yang melengkapi uraian pada bagian isi.

BAB II

TINJAUAN PUSTAKA

2.1. Teori Graf

2.1.1. Pengertian Graf

Sebuah graf G berisikan dua himpunan yaitu himpunan berhingga tak kosong V(G) dari objek-objek yang disebut titik dan himpunan berhingga (mungkin kosong) E(G) yang elemen-elemennya disebut sisi sedemikian hingga setiap elemen e dalam E(G) merupakan pasangan tak berurutan dari titik-titik di V(G). Himpunan V(G) disebut himpunan titik G dan himpunan E(G) disebut himpunan sisi G (Budayasa, 2007: 1-2).

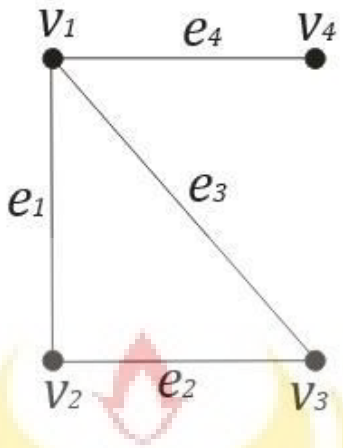
Setiap garis berhubungan dengan satu atau dua titik. Titik-titik tersebut dinamakan titik ujung. Garis yang hanya berhubungan dengan satu titik ujung disebut *loop*. Dua garis berbeda yang menghubungan titik yang sama disebut garis paralel (Siang, 2002: 186).

Cara merepresentasikan sebuah graf yang paling umum adalah dengan diagram. Dalam diagram tersebut, titik-titik dinyatakan sebagai noktah dan tiap sisi dinyatakan sebagai kurva sederhana yang menghubungkan tiap dua titik.

Contoh 2.1.

Sebuah graf G = (V,E) dengan V=$\{v_1, v_2, v_3, v_4\}$ dan E=$\{e_1, e_2, e_3, e_4\}$. $e_1 = v_1, v_2;\ e_2 = v_2, v_3; e_3 = v_1, v_3;\ e_4 = v_1, v_4$.

Graf G terlihat pada Gambar 2.1.



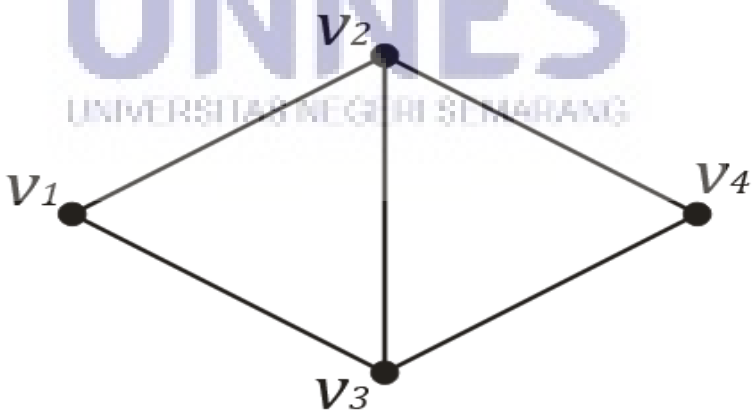
Gambar 2.1 Contoh Graf G

2.1.2. Jenis-Jenis Graf

Berdasarkan keberadaan *loop* dan sisi ganda, graf digolongkan menjadi dua jenis sebagai berikut.

2.1.2.1. Graf Sederhana *(Simple Graph)*

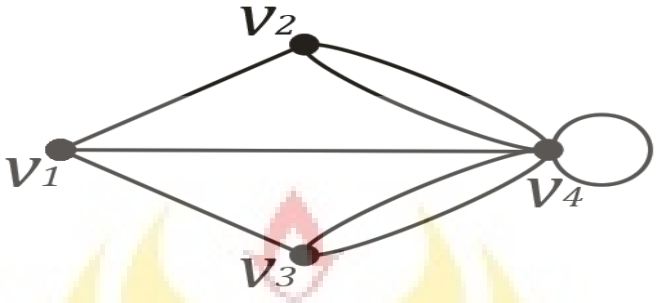
Graf sederhana (*simple graph*) adalah graf yang tidak mengandung *loop* dan sisi ganda. Contoh graf sederhana dapat dilihat pada Gambar 2.2.



Gambar 2.2. Graf Sederhana

2.1.2.2. Graf Tak-Sederhana (*Unsimple Graph*)

Graf tak-sederhana (*unsimple graph*) adalah graf yang mengandung *loop* atau sisi ganda. Contoh graf tak-sederhana dapat dilihat pada Gambar 2.3.

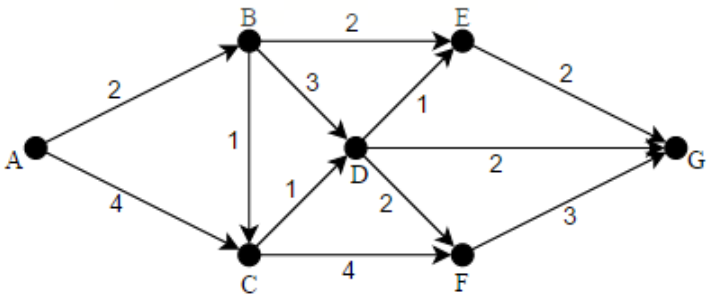


Gambar 2.3. Graf-Tak Sederhana

Berdasarkan arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu sebagai berikut.

2.1.2.3. Graf Berarah dan Berbobot

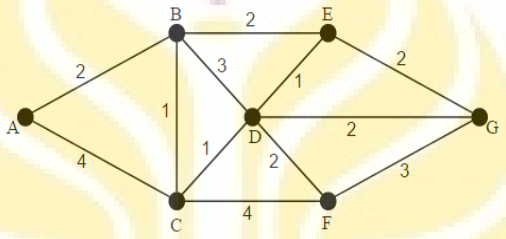
Graf berarah dan berbobot setiap sisinya mempunyai anak panah dan berbobot. Gambar 2.4 menunjukkan graf berarah dan berbobot yang terdiri atas tujuh titik yaitu titik A, B, C, D, E, F dan G. Titik A menunjukkan arah ke titik B dan titik C. Titik B menunjuk ke arah titik C, D dan titik E, dan seterusnya. Bobot antara titik A ke B adalah 2, titik A ke C adalah 4 dan seterusnya. Contoh graf berarah dan berbobot dapat dilihat pada Gambar 2.4.



Gambar 2.4. Graf Berarah dan Berbobot

2.1.2.4. Graf Tidak Berarah dan Berbobot

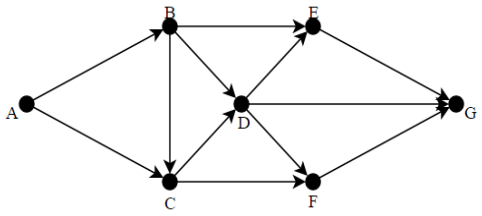
Graf tidak berarah dan berbobot setiap sisinya tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 2.5 menunjukkan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A, B, C, D, E, F, dan G. Titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain. Contoh graf tidak berarah dan berbobot dapat dilihat pada Gambar 2.5.



Gambar 2.5. Graf Tidak Berarah dan Berbobot

2.1.2.5. Graf Berarah dan Tidak Berbobot

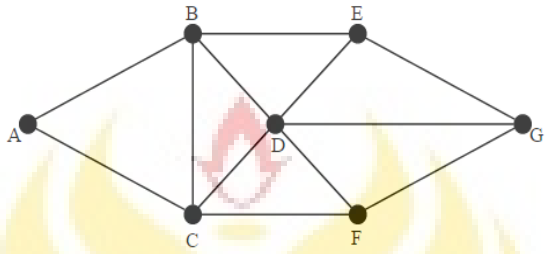
Graf berarah dan tidak berbobot setiap sisinya mempunyai anak panah yang tidak berbobot. Gambar 2.6 menunjukkan graf berarah dan tidak berbobot. Graf terdiri dari tujuh titik yaitu titik A, B, C, D, E, F, dan titik G. Titik A menunjukkan arah ke titik B atau titik C. Titik B menunjuk ke arah titik C, D dan titik E, dan seterusnya. Contoh graf berarah dan tidak berbobot dapat dilihat pada Gambar 2.6.



Gambar 2.6. Graf Berarah dan Tidak Berbobot

2.1.2.6. Graf Tidak Berarah dan Tidak Berbobot

Graf tidak berarah dan tidak berbobot setiap sisinya tidak mempunyai anak panah dan tidak berbobot. Contoh graf tidak berarah dan tidak berbobot dapat dilihat pada Gambar 2.7.



Gambar 2.7. Graf Tidak Berarah dan Tidak Berbobot

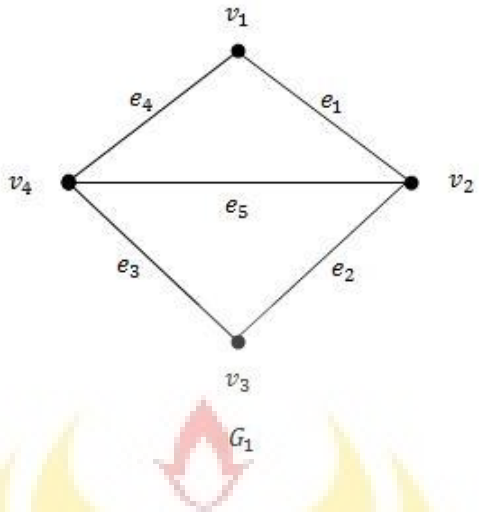
Sebuah struktur graf bisa dikembangkan dengan memberi bobot atau nilai pada tiap *edge* di mana merupakan suatu nilai yang dapat berupa biaya, jarak atau waktu, graf semacam ini disebut graf berbobot (*weighted graph*).

2.1.3. Terminologi Graf

Dalam pembahasan mengenai graf biasanya sering menggunakan terminologi (istilah) yang berkaitan dengan graf. Terminologi yang berkaitan dengan graf yaitu sebagai berikut.

2.1.3.1. Bertetangga

Dua buah titik pada graf tak berarah G dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi. Dengan kata lain v_j bertetangga dengan v_k jika (v_j, v_k) adalah sebuah sisi pada graf G (Munir, 2005: 365). Contoh graf bertetangga dapat dilihat pada Gambar 2.8.

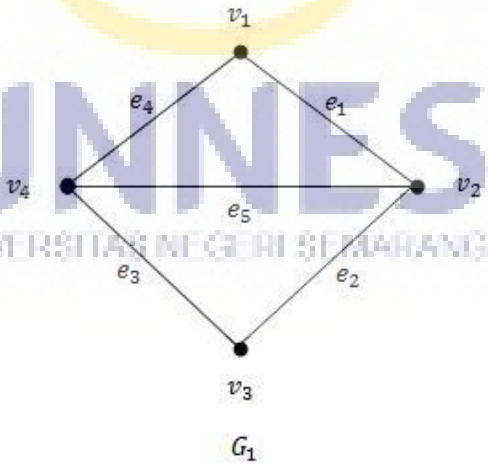


Gambar 2.8. Graf Bertetangga

Titik v_1 bertetangga dengan titik v_2 dan v_4, titik v_1 tidak bertetangga dengan titik v_3.

2.1.3.2. Bersisian *(Incident)*

Untuk sembarang sisi $e = (v_j, v_k)$, sisi e dikatakan bersisian dengan titik v_j dan titik v_k (Munir, 2005: 365). Contoh graf bersisian terlihat pada Gambar 2.9.



Gambar 2.9. Graf Bersisian

2.1.3.3. Graf Kosong *(Null Graph)*

Graf yang himpunan sisinya merupakan himpunan kosong disebut graf kosong dan ditulis sebagai N_n, n adalah jumlah titik (Munir, 2005: 366). Contoh graf kosong dapat dilihat pada Gambar 2.10.

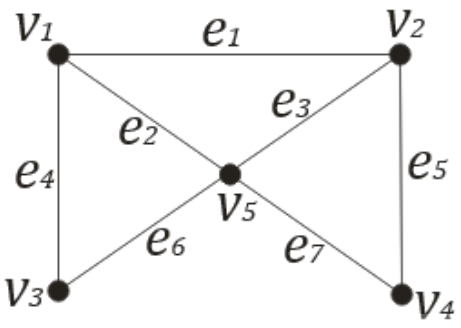
Gambar 2.10. Graf Kosong

2.1.3.4. Derajat *(Degree)*

Derajat suatu titik pada graf tak berarah adalah jumlah sisi yang bersisian dengan titik tersebut (Munir, 2005: 366).

2.1.3.5. Jalan *(Walk)*

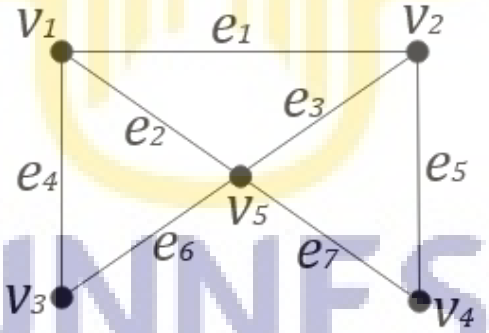
Misalkan *G* adalah sebuah graf. Sebuah jalan (*walk*) di *G* adalah sebuah barisan berhingga (tak kosong), $W = (v_0, e_1, v_1, e_2, v_2, \dots, e_k, v_k)$ yang suku-sukunya bergantian sedemikian hingga v_{i-1} dan v_i adalah titik-titik akhir sisi e_i untuk $1 \leq i \leq k$. Dikatakan *W* adalah sebuah jalan dari titik v_0 ke titik v_k, atau jalan (v_0, v_k). Titik v_0 dan titik v_k berturut-turut disebut titik awal dan titik akhir *W*. Sedangkan titik-titik $v_1, v_2, \dots, v_{k-1}$ disebut titik internal *W*, dan *k* disebut panjang jalan *W*. Panjang jalan *W* adalah banyaknya sisi dalam *W*. Sebuah jalan *W* dengan panjang positif disebut tertutup, jika titik awal dan titik akhir dari *W* identik (sama) (Budayasa, 2007:6). Jalan terlihat pada Gambar 2.11.



Gambar 2.11. Jalan pada Graf *G* ($v_3 - v_5 - v_4 - v_2 - v_1 - v_3$)

2.1.3.6. Jejak (*Trail*)

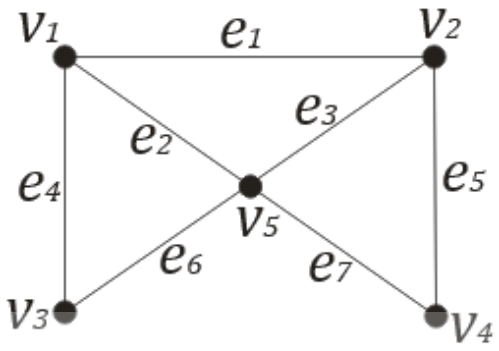
Sebuah titik *G*, mungkin saja muncul lebih dari satu kali dalam jalan *W*, begitu juga dengan sebuah sisi *G*, boleh muncul lebih dari satu kali pada jalan *W*. Jika semua sisi $e_1, e_2, e_3, \ldots, e_k$ dalam jalan *W* berbeda, maka *W* disebut sebuah jejak (*Trail*) (Budayasa,2007:6). Jejak terlihat pada Gambar 2.12.



Gambar 2.12. Jejak pada Graf *G* ($v_3 - v_5 - v_2 - v_1 - v_5 - v_4$)

2.1.3.7. Lintasan *(Path)*

Jika semua titik $v_0, v_1, v_2, \ldots, v_k$ dalam jejak *W* berbeda, maka *W* disebut lintasan (*Path*) (Budayasa, 2007:6). Lintasan terlihat pada Gambar 2.13.



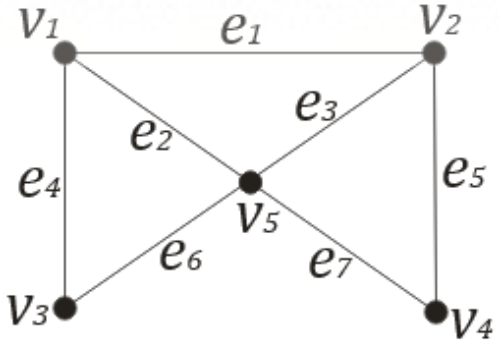
Gambar 2.13. Lintasan pada Graf G ($v_1 - v_3 - v_5 - v_2 - v_4$)

2.1.3.8. Jalur *(Path)*

Jalur adalah walk yang semua titik dalam barisan adalah berbeda.

2.1.3.9. Sirkuit *(Cycle)*

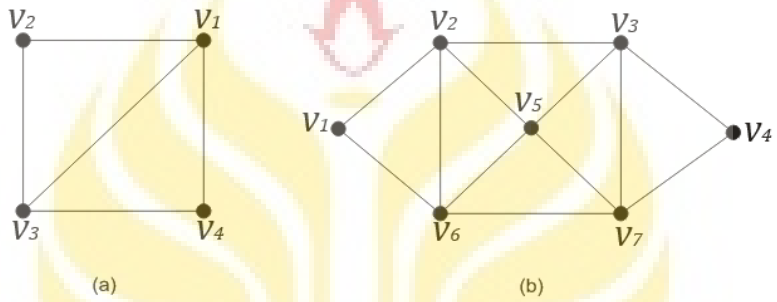
Jejak yang berawal dan berahir pada titik yang sama disebut sirkuit (Budayasa,2007:6). Sebuah sirkuit dikatakan sirkuit sederhana (*Simple Circuit*) jika sirkuit tersebut tidak memuat atau melewati sisi yang sama dua kali (setiap sisi yang dilewati hanya sekali). Sebuah sirkuit dikatakan sirkuit dasar (*Elementary Circuit*) jika sirkuit tersebut tidak memuat atau melewati titik yang sama dua kali (setiap titik yang dilewati hanya satu kali, titik awal dan titik akhir boleh sama). Sirkuit terlihat pada Gambar 2.14.



Gambar 2.14. Sirkuit pada Graf G ($v_1 - v_3 - v_5 - v_4 - v_2 - v_1$)

2.1.4. Graf Euler dan Graf Semi-Euler

Sebuah sirkuit di graf *G* yang memuat semua sisi *G* disebut sirkuit *Euler*. Jika graf *G* memuat semua sirkuit *Euler*, maka graf *G* disebut Graf *Euler*. Sebuah jejak buka yang memuat semua sisi graf disebut jejak *Euler*. Graf *G* disebut graf semi-*euler* jika *G* memuat jejak *Euler* (Budayasa,2007:113). Eulerian pada Graf terlihat pada Gambar 2.15.



Gambar 2.15. Eulerian pada Graf

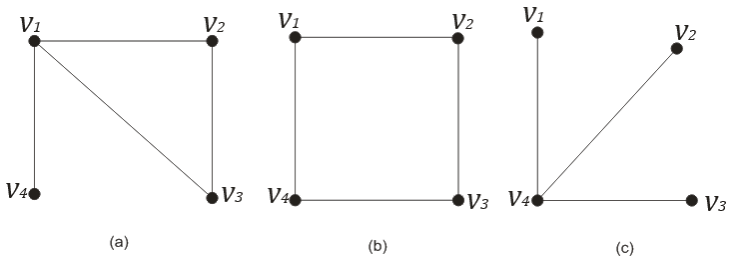
Keterangan:

(a) Graf *G* memiliki jejak *Euler*, yaitu $v_3 - v_1 - v_2 - v_3 - v_4 - v_1$

(b) Graf *G* memiliki Sirkuit *Euler*, yaitu $v_1 - v_2 - v_3 - v_4 - v_7 - v_3 - v_5 - v_7 - v_6 - v_5 - v_2 - v_6 - v_1$

2.1.5. Lintasan dan Sirkuit Hamilton

Misalkan *G* sebuah graf. Sebuah lintasan di *G* yang memuat semua titik di *G* disebut Lintasan Hamilton. Graf non Hamilton yang memuat lintasan hamilton disebut graf semi-hamilton (Budayasa, 2007:130). Sirkuit hamilton adalah sirkuit yang melalui tiap titik di graf tepat satu kali, kecuali titik asal (sekaligus titik akhir) yang dilalui dua kali. Graf Hamilton adalah graf yang memiliki sirkuit hamilton (Munir, 2010:232). Hamiltonian pada Graf terlihat pada Gambar 2.16.



Gambar 2.16. Hamiltonian pada Graf *G*

Keterangan:

(a) Graf *G* memiliki lintasan hamilton $v_3 - v_2 - v_1 - v_4$

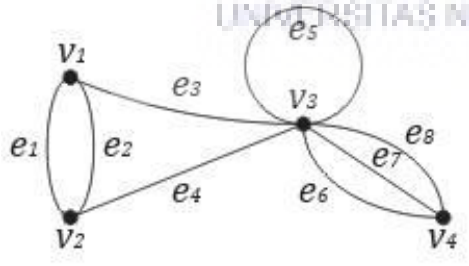
(b) Graf *G* memiliki Sirkuit Hamilton $v_1 - v_2 - v_3 - v_4 - v_1$

(c) Graf *G* tidak memiliki Lintasan maupun Sirkuit Hamilton

2.1.6. Representasi Graf

2.1.6.1. Matriks Berhubungan Langsung *(Adjacency Matrix)*

Misalkan G sebuah Graf dengan $v(G) = \{v_1, v_2, \ldots., v_n\}$. Matriks berhubungan langsung graf G adalah matriks bujur sangkar $A(G) = (a_{ij})$ berordo nxn yang baris-baris dan kolom-kolomnya diberi label titik-titik graf G yang sedemikian hingga elemen a_{ij} menyatakan banyaknya sisi G yang menghubungkan titik v_1 dan v_j.



$$A(G) = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{cccc} V_1 & V_2 & V_3 & V_4 \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & 1 & 1 & 3 \\ 0 & 0 & 3 & 0 \end{bmatrix} \end{array}$$

Gambar 2.17. Graf G dan Matriks Berhubungan Langsungnya

2.1.6.2. *Adjancency List*

Selain dengan matriks berhubungan langsung, sebuah graf dapat disajikan dalam matriks keterkaitan. Misalkan graf G mempunyai n buah titik $v_1, v_2, \ldots, v_n$ dan t buah sisi $e_1, e_2, \ldots, e_t$. Matriks keterkaitan *(incidence matrix)* graf G adalah matriks $M(G) = (m_{ij})$ berordo nxt yang baris-barisnya dilabel dengan label titik-titik G dan kolom-kolomnya dilabel dengan label sisi-sisi G, sedemikian hingga

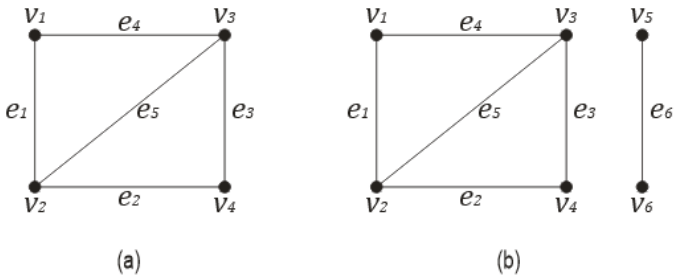
$$m_{ij} = \begin{cases} 0 & , \text{ jika sisi } e_j \textit{ tidak terkait dengan titik } V_i \\ 1 & , \textit{ jika sisi } e_j \textit{ terkait dengan titik } V_i \textit{ dan } e_j \textit{ bukan gelung} \\ 2 & , \textit{ jika sisi } e_j \textit{ terkait dengan titik } V_i \textit{ dan } e_j \textit{ gelung} \end{cases}$$

Perhatikan graf G pada Gambar 2.17. Matriks keterkaitan graf tersebut adalah sebagai berikut.

$$A = \begin{array}{c} \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{array} \begin{array}{c} \begin{array}{cccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 2 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

2.1.7. Graf Terhubung (*Connected Graph*)

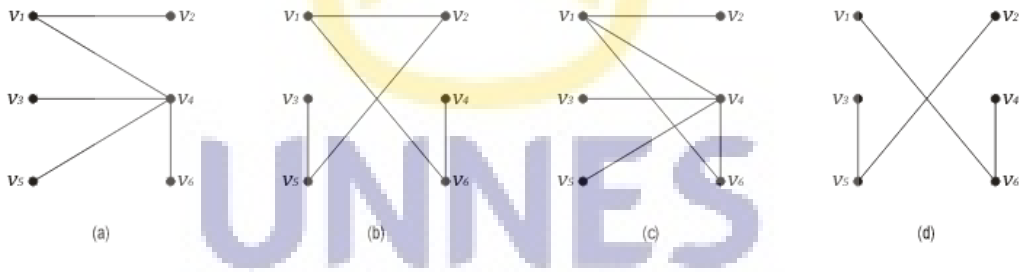
Suatu graf G dikatakan terhubung (*connected*) jika untuk setiap dua titik G yang berbeda terdapat sebuah lintasan yang menghubungan kedua titik tersebut. Sebaliknya graf G adalah sebuah graf bagian terhubung maksimal (titik dan sisi) dari G (Budayasa, 2007: 8). Graf H bagian terhubung maksimal dari graf G apabila tidak ada graf bagian lain dari G yang terhubung dan memuat H. Jadi setiap graf terhubung memiliki tepat satu komponen sedangkan graf tidak terhubung memiliki paling sedikit dua komponen. Graf terhubung terlihat pada Gambar 2.18.



Gambar 2.18. (a) Graf Terhubung, (b) Graf Tak Terhubung

2.1.8. Pohon

Misalkan G adalah suatu graf sederhana (tidak memiliki garis paralel dan *loop*). G disebut pohon bila dan hanya bila tidak memuat sirkuit dan terhubung. Pohon semu (*trivial tree)* adalah pohon yang hanya terdiri dari sebuah titik. Pohon kosong (*empty tree)* adalah pohon yang tidak mempunyai titik. G disebut hutan *(forest)* bila dan hanya bila G tidak memuat sirkuit (Siang, 2011:279). Contoh pohon terlihat pada Gambar 2.19.



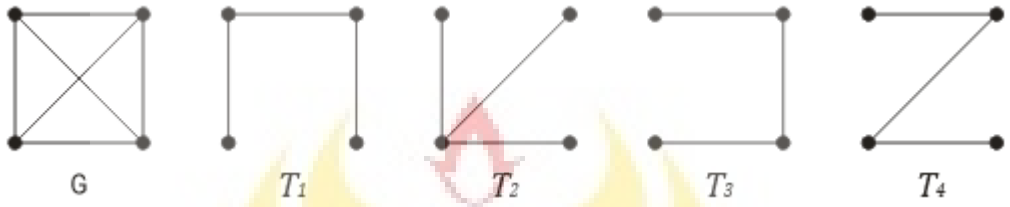
Gambar 2.19 Contoh Graf Pohon dan Bukan Pohon

Keterangan gambar sebagai berikut.

1. Gambar 2.19. (a) merupakan pohon karena terhubung dan bukan sirkuit.
2. Gambar 2.19. (b) merupakan pohon karena terhubung dan bukan sirkuit.
3. Gambar 2.19. (c) bukan pohon karena terdapat sirkuit v_1, v_4, v_6, v_1
4. Gambar 2.19. (d) bukan pohon karena v_3, v_5, v_2 dan v_1, v_6, v_4 bukan graf terhubung.

2.1.8.1. Pohon Rentang *(Spanning Tree)*

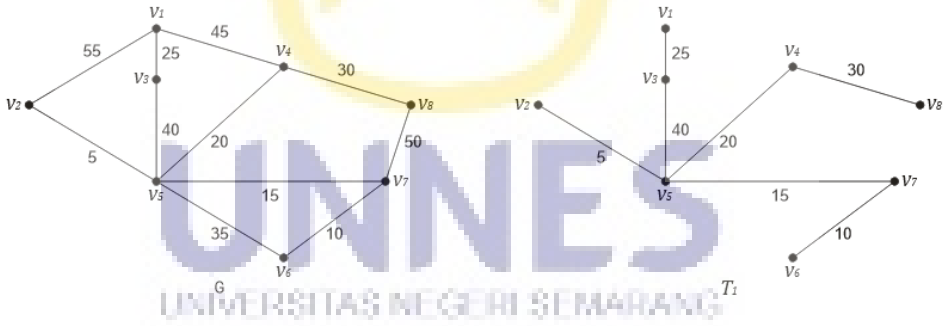
Misalkan G adalah sebuah graf. Sebuah pohon di G yang memuat semua titik G disebut pohon rentang *(spanning tree)* (Budayasa,2007:32). Contoh pohon rentang terlihat pada Gambar 2.20.



Gambar 2.20. Contoh Pohon-Pohon Rentang pada Graf G

2.1.8.2. Pohon Rentang Minimum *(Minimum Spanning Tree)*

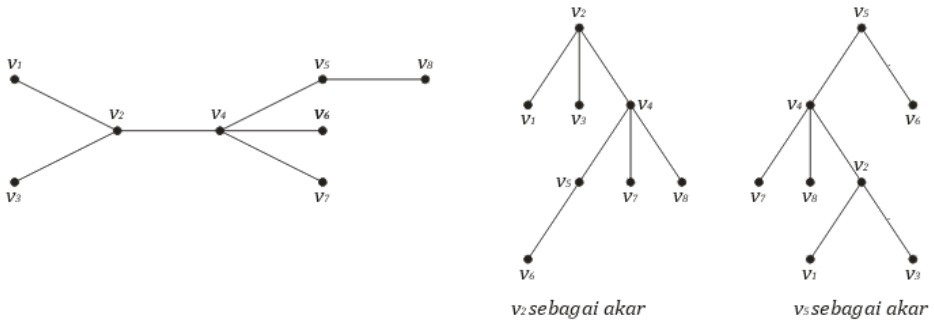
Misalkan G adalah graf berlabel. Pohon rentang minimum adalah pohon rentang G dengan total bobot seminimum mungkin (Siang,2011:282). Contoh pohon rentang minimum terlihat pada Gambar 2.21.



Gambar 2.21. T_1 Pohon Rentang Minimum dari Graf G

2.1.8.3. Pohon Berakar

Pohon yang satu buah titiknya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar *(rooted tree).* Pohon dan dua buah pohon berakar yang dihasilkan dari pemilihan dua titik berbeda sebagai akar. Contoh pohon Berakar terlihat pada Gambar 2.22.

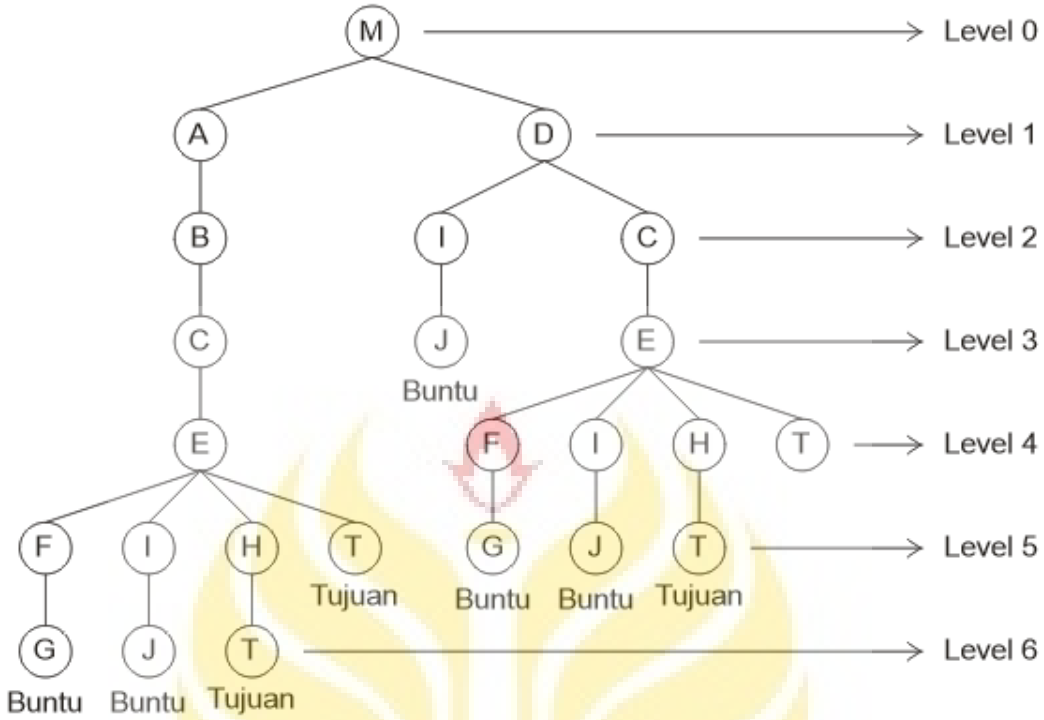


Gambar 2.22. Macam-Macam Pohon Berakar

2.1.8.4. Pohon Ruang Status

Untuk menghindari kemungkinan adanya proses pencarian titik berulang maka digunakan struktur pohon, struktur pohon inilah yang kemudian disebut sebagai pohon ruang status. Pohon ruang status digunakan untuk menggambarkan keadaan secara sistematis. Pohon terdiri atas titik-titik. Titik yang terletak pada level 0 disebut akar. Titik akar menunjukkan keadaan awal yang biasanya merupakan topik atau objek. Titik akar memiliki beberapa percabangan yang terdiri atas beberapa titik *successor* yang disebut anak. Titik yang memiliki beberapa titik anak disebut juga dengan titik induk (orang tua titik).

Titik yang tidak memiliki anak disebut daun yang menunjukkan akhir dari suatu pencarian, dapat berupa tujuan yang diharapkan (*goal*) atau jalan buntu (*dead end*).



Gambar 2.23 Pohon Ruang Status

Keterangan

Titik akar : M

Titik induk (orangtua titik) : M, A, B, C, D, E, F, H, dan I

Titik anak : A, B, C, E, F, G ,H, I, J, dan T.

Daun : -T (tujuan)

-G dan J (buntu)

2.1.9. Lintasan Terpendek

Lintasan terpendek merupakan salah satu dari masalah yang dapat diselesaikan dengan graf. Jika diberikan sebuah graf berbobot, masalah lintasan terpendek adalah bagaimana cara mencari sebuah jalur pada graf yang dapat meminimalkan jumlah bobot sisi pembentuk jalur tersebut. Beberapa macam persoalan lintasan terpendek adalah sebagai berikut.

1. Lintasan terpendek antara dua buah titik tertentu (*a pair shortest path*)
2. Lintasan terpendek antara semua pasangan titik (*all pairs shortest path*)
3. Lintasan terpendek dari titik tertentu ke semua titik yang lain.
4. Lintasan terpendek antara dua buah titik yang melalui beberapa titik tertentu (*intermediate shortest path*)

Panjang lintasan dalam sebuah graf berbobot adalah jumlah bobot semua sisi pada lintasan tersebut. Misalkan u dan v dua titik di graf G. Lintasan$-(u,v)$ di G dengan panjang minimum disebut lintasan terpendek antara u dan v. Jarak dari u ke v, dinotasikan dengan $d(u,v)$ atau $d(v,u)$ didefinisikan sebagai panjang lintasan terpendek antara titik u dan titik v di G (Budayasa, 2007:47).

2.2. *Travelling Salesman Problem* (TSP)

Travelling Salesman Problem termasuk ke dalam persoalan yang sangat terkenal dalam teori graf. Nama persoalan ini diilhami oleh masalah seorang pedagang yang akan mengunjungi sejumlah kota. Uraian persoalannya adalah sebagai berikut.

Pedagang menggunakan waktunya untuk mengunjungi n kota (*nodes*) secara siklus perputaran. Di dalam satu kali perjalanan keliling, ia harus menentukan urutan dari sejumlah kota yang harus dilaluinya, setiap kota hanya boleh dilalui sekali dan hanya sekali dalam perjalanan, dan perjalanan berakhir pada kota awal di mana ia memulai perjalanan.

Kebanyakan *Travelling Salesman Problem* merupakan suatu simetris yang berarti untuk dua kota A dan B, jarak dari kota A ke kota B adalah sama dengan

jarak dari kota B ke kota A. Dalam hal ini, kita akan mendapatkan panjang perjalanan keliling yang sama persis jika kita membalikkan sirkuit perjalanan tersebut. Jadi tidak ada perbedaan antara suatu perjalanan keliling dan kebalikannya.

2.2.1. Sejarah *Travelling Salesman Problem*

Permasalahan matematik yang berkaitan dengan *Travelling Salesman Problem* mulai muncul sekitar tahun 1800-an. Masalah ini dikemukakan oleh dua orang matematikawan, yaitu Sir William Rowan Hamilton yang berasal dari Irlandia dan Thomas Penyngton Kirkman yang berasal dari Inggris. Diskusi mengenai awal studi dari persoalan TSP ini dapat ditemukan di buku *Graph Theory* 1736-1936 by N.L. Biggs, E.K. LLoyd, and R.J. Wilson, Clarendon Press, Oxford, 1976. Bentuk umum dari persoalan TSP pertama kali dipelajari oleh para matematikawan mulai tahun 1930-an Karl Menger di Vienna dan Harvard. Persoalan tersebut kemudian dikembangkan oleh Hassler Whitney dan Merril Flood di Princeton. Penelitian secara mendetail hubungan antara Menger dan Whitney, dan perkembangan persoalan TSP sebagai sebuah topik studi dapat ditemukan pada tulisan Alexander Schriver "*On the history of combinatorial optimization (till 1960)*"(Zulfikar,2008:5).

2.2.2. Contoh Perkembangan Masalah yang Muncul

Kode program komputer yang dibuat untuk menyelesaikan persoalan TSP telah berkembang semakin baik dari tahun ke tahun. Tanda yang paling mencolok dari perkembangan metode untuk menyelesaikan persoalan TSP adalah

bertambahnya jumlah simpul (*node*) yang dapat diselesaikan, mulai dari solusi persoalan 49 kota yang dikembangkan oleh Dantzig, Fulkerson, dan Johnson's pada tahun 1954 sampai kepada solusi persoalan 24.978 kota pada tahun 2004, data-data tersebut didapat dari National Imagery and Mapping Agency, sebuah *database* nasional yang menyimpan nama-nama fitur geografi (Zulfikar,2008:5).

2.3. Algoritma *Branch and Bound*

Pemecahan masalah optimasi *Travelling Salesman Problem* merupakan pekerjaan yang membutuhkan algoritma yang efisien dan algoritma *Branch and Bound* merupakan salah satu algoritma untuk memecahkan masalah optimasi *Travelling Salesman Problem*.Algoritma *Branch and Bound* mencari sejumlah solusi yang lengkap untuk masalah yang ada dengan hasil yang terbaik. Walaupun begitu, penggunaan satu per satu secara eksplisit tidak mungkin dilakukan dalam kaitan penambahan sejumlah solusi yang potensial. Penggunaan batas *(bound)* untuk fungsi yang akan dioptimalkan dikombinasikan dengan nilai solusi terbaik yang ada memungkinkan algoritma untuk mencari bagian-bagian dari sejumlah solusi secara implisit.

Algoritma *Branch and Bound* merupakan metode pencarian di dalam ruang solusi secara sistematis (Widyawati, Mashuri & Arifudin, 2014:45). *Branch and Bound* adalah suatu yang membagi (*divide*) dan menaklukan (*conquer*) yang mengurangi masalah asli untuk satu rangkaian lebih kecil dari sub persoalan dan kemudian secara berulang memecahkan masalah masing-masing sub persoalan tersebut. Ada tiga komponen dalam algoritma *Branch and Bound*, yaitu sebagai berikut.

1. Fungsi Pembatas (*Bounding*) : fungsi yang disediakan *subspace* dari ruang solusi dengan batas rendah untuk nilai solusi terbaik yang diperoleh dalam *subspace.* Metode dalam bounding ada dua, yaitu sebagai berikut.

   (a) Metode *upper bounding* : suatu metode untuk menentukan suatu batas atas pada solusi optimal.

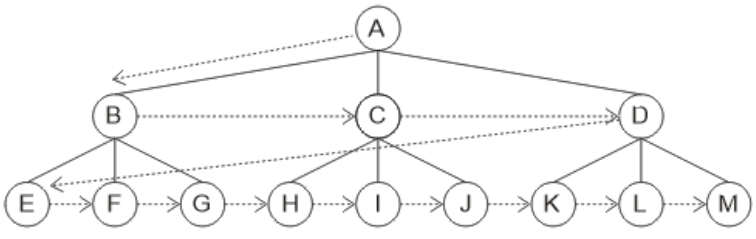
   (b) Metode *lower bounding* : suatu metode untuk menentukan suatu batas bawah dari fungsi objektif.

2. Strategi pencarian : suatu strategi untuk menyeleksi tiap-tiap node yang dihasilkan dan mendapatkan node yang optimum.

3. Metode percabangan (*branching*) : suatu metode yang diaplikasikan jika subspace setelah diperiksa tidak dapat dibatalkan, karena itu pembagian *subspace* kedalam dua atau lebih subspace untuk diperiksa dalam sub rangkaian iterasi.

Algoritma *Branch and Bound* adalah algoritma pencarian di dalam ruang solusi secara sistematis. Ruang solusi diorganisasikan ke dalam pohon ruang status. Pohon ruang status dibangun berdasarkan skema BFS (*Breadth Firs Searcht*) (Riyanto,2014).

BFS merupakan metode pencarian melebar dalam pohon ruang status secara transversal yang dimulai dari titik akar terus ke level 1 dari titik kiri ke kanan, kemudian berpindah ke level berikutnya sampai ditemukan titik tujuan (solusi).



Gambar 2.24 Graf *Breadth First Search*

Pada algoritma *Branch and Bound*, pencarian ke titik solusi dapat dipercepat dengan memilih titik hidup berdasarkan nilai ongkos (*cost*). Setiap titik hidup diasosiasikan dengan sebuah ongkos yang menyatakan nilai bebas (*bound*). Batas ini dapat berupa batas bawah (*lower bound*) ataupun batas atas (*upper bound*) masing-masing menyatakan batas maksimum atau batas minimum yang diperlukan untuk membangkitkan sebuah titik.

Pemberian nilai batas akan ideal jika diketahui letak titik solusi, namun kebanyakan persoalan letak titik solusi tidak diketahui sehingga nilai batas untuk setiap titiknya umumnya hanya berupa taksiran ataupun perkiraan. Untuk membantu pemberian taksiran nilai, umumnya digunakan fungsi heuristik yang dapat berupa fungsi apapun asalkan fungsi tersebut mampu memodelkan ongkos (*cost*) untuk membantu mencapai titik solusi. Fungsi heuristik untuk menghitung taksiran nilai tersebut dinyatakan secara umum sebagai berikut.

$$\hat{c}(S) = \hat{c}(R) + A(i,j) + r$$

Keterangan sebagai berikut.

$\hat{c}(S)$ = bobot perjalanan yang melalui titik S (titik di pohon ruang status).

$\hat{c}(R)$ = bobot perjalanan minimum yang melalui titik R, dalam hal ini R adalah orang tua dari S.

$A(i,j)$ = bobot sisi (i,j) pada graf G yang berkorespondensi dengan sisi (R,S) pada pohon ruang status.

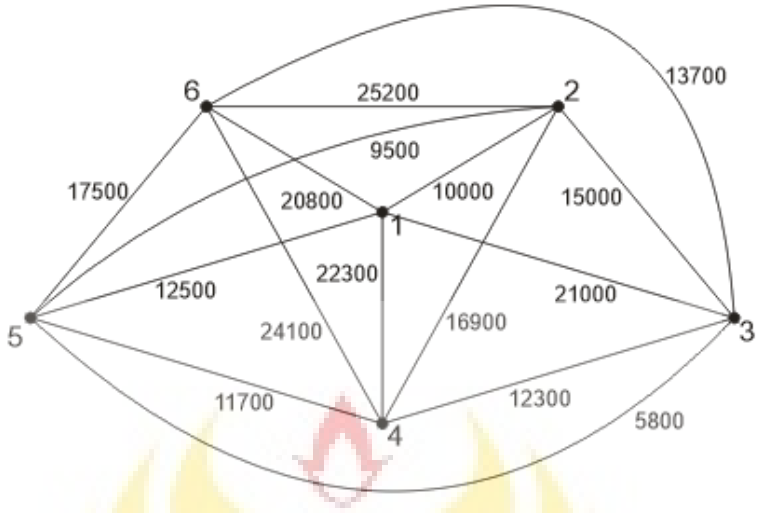
r = jumlah semua pengurangan pada proses memperoleh matriks tereduksi untuk titik S.

Nilai $\hat{c}$ digunakan untuk mengurutkan pencarian. Jika pencarian adalah pencarian dengan nilai biaya terkecil maka titik berikutnya dipilih untuk diekspansi adalah titik yang memiiki $\hat{c}$ minimum. Bila sebaliknya, titik yang diekspansi adalah titik yang memiliki $\hat{c}$ maksimum.

Skema umum algoritma *branch and bound* adalah sebagai berikut.

1. Masukkan titik akar ke dalam antrian Q. Jika titik akar adalah titik solusi (*goal node*), maka solusi telah ditemukan. Stop
2. Jika Q kosong, tidak ada solusi. Stop
3. Jika Q tidak kosong, pilih dari antrian Q titik i yang mempunyai $\hat{c}(i)$ paling kecil. Jika terdapat beberapa titik i yang memenuhi, pilih satu secara sembarang.
4. Jika titik i adalah titik solusi, berarti solusi ditemukan, stop. Jika titik i bukan titik solusi, maka bangkitkan semua titik anaknya. Jika i tidak mempunyai titik anak, kembali ke langkah 2.
5. Untuk setiap titik anak j dari titik i, hitung j, dan masukkan semua titik-titik anak tersebut ke dalam antrian Q.
6. Kembali ke langkah 2.

Berikut ini adalah contoh penerapan *algoritma Branch and Bound* untuk memecahkan suatu permasalahan.



Gambar 2.25 Graf yang Menggambarkan Posisi Titik-Titik Tujuan

Keterangan Gambar 2.25 sebagai berikut.

Titik 1 mewakili *start*, titik 2 adalah titik tujuan T2, titik 3 adalah titik tujuan T3, titik 4 adalah titik tujuan T4, titik 5 adalah titik tujuan T5, dan titik 6 adalah titik tujuan T6.

Matriks 6x6 di mana elemen M_{ij} adalah jarak dari i ke j, sedangkan i dan j adalah titik pada graf.

$$\begin{bmatrix} \infty & 10000 & 21000 & 22300 & 12500 & 20800 \\ 10000 & \infty & 15000 & 16900 & 9500 & 25200 \\ 21000 & 15000 & \infty & 12300 & 5800 & 13700 \\ 22300 & 16900 & 12300 & \infty & 11700 & 24100 \\ 12500 & 9500 & 5800 & 11700 & \infty & 17500 \\ 20800 & 25200 & 13700 & 24100 & 17500 & \infty \end{bmatrix}$$

Matriks di atas merupakan matriks simetris karena jarak i ke j sama dengan jarak j ke i. Dalam penyederhanaan matrik di atas dilakukan dengan cara mereduksi matriks tersebut.

Reduksi baris

$$\begin{bmatrix} \infty & 0 & 11000 & 12300 & 2500 & 10800 \\ 500 & \infty & 5500 & 7400 & 0 & 15700 \\ 15200 & 9200 & \infty & 6500 & 0 & 7900 \\ 10600 & 5200 & 600 & \infty & 0 & 12400 \\ 6700 & 3700 & 0 & 5900 & \infty & 11700 \\ 7100 & 11500 & 0 & 10400 & 3800 & \infty \end{bmatrix}$$

Matriks di atas dihasikan dari pengurangan tiap baris dengan nilai terkecil pada elemen baris tersebut. Baris ke-1 dikurangi 10000, baris ke-2 dikurangi 9500, baris ke-3 dikurangi 5800, baris ke-4 dikurangi 11700, baris ke-5 dikurangi 5800, dan baris ke-6 dikurangi 13700.

Reduksi kolom

$$\begin{bmatrix} \infty & 0 & 11000 & 6400 & 2500 & 2900 \\ 0 & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix}$$

Matriks di atas dihasilkan dari pengurangan seluruh elemen pada kolom 1 dengan 500, kolom 4 dengan 5900, dan kolom 5 dengan 7900. Selanjutnya, proses reduksi ini akan menghasilkan nilai batas titik akar atau $\hat{c}(root)$, yang didapat dari penjumlahan semua elemen pengurangan tersebut. Jadi $\hat{c}(root) = 10000 + 9500 + 5800 + 11700 + 5800 + 13700 + 500 + 5900 + 7900 = 70800$.

Dengan demikian berarti telah dibangkitkan pohon ruang status yang baru berisi satu titik dengan bobot 70800. Pohon ruang status level 0 terlihat pada Gambar 2.26.

Gambar 2.26 Pohon Ruang Status Level 0

Selanjutnya menghitung titik-titik lain pada pohon ruang status dengan mangacu pada persamaan yang telah dijelaskan di atas.

1. Titik 2; lintasan 1,2

$$\begin{bmatrix} \infty & 0 & 11000 & 6400 & 2500 & 2900 \\ 0 & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & \infty & \infty & 600 & 0 & 0 \\ 10100 & \infty & 600 & \infty & 0 & 4500 \\ 6200 & \infty & 0 & 0 & \infty & 3800 \\ 6600 & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 5500 & 1500 & 0 & 7800 \\ 8500 & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & 600 & \infty & 0 & 4500 \\ 0 & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan kolom ke-1 oleh 6200

$$\hat{c}(2) = 7080 + 0 + 6200 = 77000$$

2. Titik 3; lintasan 1,3

$$\begin{bmatrix} \infty & 0 & 11000 & 6400 & 2500 & 2900 \\ 0 & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 1500 & 0 & 7800 \\ \infty & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & \infty & \infty & 0 & 4500 \\ 6200 & 3700 & \infty & 0 & \infty & 3800 \\ 6600 & 11500 & \infty & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 1500 & 0 & 7800 \\ \infty & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & \infty & \infty & 0 & 4500 \\ 6200 & 3700 & \infty & 0 & \infty & 3800 \\ 2800 & 7700 & \infty & 700 & 0 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 1500 & 0 & 7800 \\ \infty & 5500 & \infty & 600 & 0 & 0 \\ 10100 & 1500 & \infty & \infty & 0 & 4500 \\ 6200 & 0 & \infty & 0 & \infty & 3800 \\ 2800 & 4000 & \infty & 700 & 0 & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan baris ke-6 oleh 3800 dan kolom ke-2 oleh 3700. $\hat{c}(3) = 70800 + 11000 + 3800 + 3700 = 89300$

3. Titik 4; lintasan 1,4

$$\begin{bmatrix} \infty & 0 & 11000 & 6400 & 2500 & 2900 \\ 0 & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & \infty & 0 & 7800 \\ 14700 & 9200 & \infty & \infty & 0 & 0 \\ \infty & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & \infty & \infty & 3800 \\ 6600 & 11500 & 0 & \infty & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & \infty & 0 & 7800 \\ 14700 & 5500 & \infty & \infty & 0 & 0 \\ \infty & 1500 & 600 & \infty & 0 & 4500 \\ 6200 & 0 & 0 & \infty & \infty & 3800 \\ 6600 & 7800 & 0 & \infty & 3800 & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan kolom ke-2 oleh 3700. $\hat{c}(4) = 70800 + 6400 + 3700 = 80900$.

4. Titik 5; lintasan 1,5

$$\begin{bmatrix} \infty & 0 & 11000 & 6400 & 2500 & 2900 \\ 0 & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & 1500 & \infty & 7800 \\ 14700 & 9200 & \infty & 600 & \infty & 0 \\ 10100 & 5200 & 600 & \infty & \infty & 4500 \\ \infty & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & 1500 & \infty & 7800 \\ 14700 & 9200 & \infty & 600 & \infty & 0 \\ 9500 & 4600 & 0 & \infty & \infty & 3900 \\ \infty & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & 1500 & \infty & 7800 \\ 14700 & 5500 & \infty & 600 & \infty & 0 \\ 9500 & 900 & 0 & \infty & \infty & 3900 \\ \infty & 0 & 0 & 0 & \infty & 3800 \\ 6600 & 7800 & 0 & 4500 & \infty & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan baris ke-4 oleh 600 dan kolom ke-2 oleh 3700. $\hat{c}(5) = 70800 + 2500 + 600 + 3700 = 77600$.

5. Titik 6; lintasan 1,6

$$\begin{bmatrix} \infty & 0 & 11000 & 6400 & 2500 & 2900 \\ 0 & \infty & 5500 & 1500 & 0 & 7800 \\ 14700 & 9200 & \infty & 600 & 0 & 0 \\ 10100 & 5200 & 600 & \infty & 0 & 4500 \\ 6200 & 3700 & 0 & 0 & \infty & 3800 \\ 6600 & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

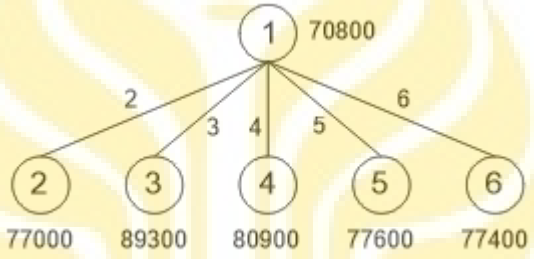
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & 1500 & 0 & \infty \\ 14700 & 9200 & \infty & 600 & 0 & \infty \\ 10100 & 5200 & 600 & \infty & 0 & \infty \\ 6200 & 3700 & 0 & 0 & \infty & \infty \\ \infty & 11500 & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 5500 & 1500 & 0 & \infty \\ 14700 & 5500 & \infty & 600 & 0 & \infty \\ 10100 & 1500 & 600 & \infty & 0 & \infty \\ 6200 & 0 & 0 & 0 & \infty & \infty \\ \infty & 7800 & 0 & 4500 & 3800 & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan kolom ke-2 oleh 3700.

$\hat{c}(6) = 70800 + 2900 + 3700 = 77400.$

Pohon ruang status yang terbentuk dari proses reduksi di atas sampai saat ini terlihat pada Gambar 2.27.



Gambar 2.27 Pohon Ruang Status Level 1

Dengan penomoran titik sama seperti sebelumnya, yaitu: (1)*start*, (2)T2, (3)T3, (4)T4, (5)T5, dan (6)T6.

Selanjutnya memilih titik yang memiliki nilai batas terkecil, dalam hal ini hanya terdapat 1 titik yaitu titik 2 berikut hasil ekspansi titik 2.

6. Titik 7; lintasan 1,2,3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 5500 & 1500 & 0 & 7800 \\ 8500 & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & 600 & \infty & 0 & 4500 \\ 0 & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & \infty & \infty & 0 & 4500 \\ 0 & \infty & \infty & 0 & \infty & 3800 \\ 400 & \infty & \infty & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & \infty & \infty & 0 & 4500 \\ 0 & \infty & \infty & 0 & \infty & 3800 \\ 0 & \infty & \infty & 4100 & 3400 & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan baris ke-6 oleh 400. $\hat{c}(7) = 77000 + 5500 + 400 = 82900$.

7. Titik 8; lintasan 1,2,4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 5500 & 1500 & 0 & 7800 \\ 8500 & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & 600 & \infty & 0 & 4500 \\ 0 & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8500 & \infty & \infty & \infty & 0 & 0 \\ \infty & \infty & 600 & \infty & 0 & 4500 \\ 0 & \infty & 0 & \infty & \infty & 3800 \\ 400 & \infty & 0 & \infty & 3800 & \infty \end{bmatrix}$$

$$\hat{c}(8) = 77000 + 1500 + 0 = 78500$$

8. Titik 9; lintasan 1,2,5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 5500 & 1500 & 0 & 7800 \\ 8500 & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & 600 & \infty & 0 & 4500 \\ 0 & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8500 & \infty & \infty & 600 & \infty & 0 \\ 3900 & \infty & 600 & \infty & \infty & 4500 \\ \infty & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8500 & \infty & \infty & 600 & \infty & 0 \\ 3300 & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & 600 & \infty & 0 \\ 2900 & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & 0 & 0 & \infty & 3800 \\ 0 & \infty & 0 & 4500 & \infty & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan baris ke-4 oleh 600 dan kolom ke-1 oleh 400. $\hat{c}(9) = 77000 + 0 + 600 + 400 = 78000$.

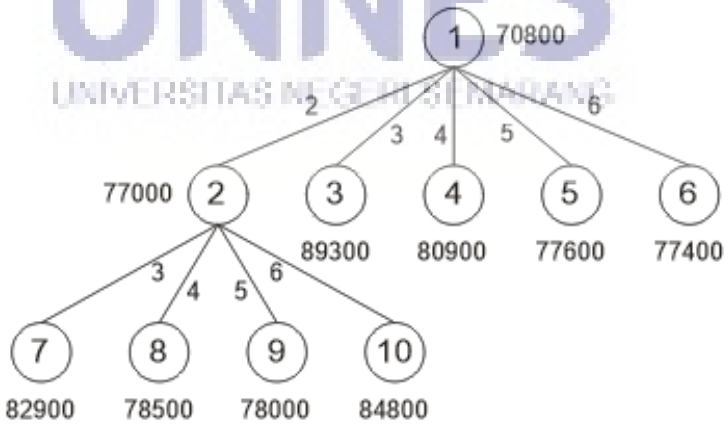
9. Titik 10; lintasan 1,2,6

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 5500 & 1500 & 0 & 7800 \\ 8500 & \infty & \infty & 600 & 0 & 0 \\ 3900 & \infty & 600 & \infty & 0 & 4500 \\ 0 & \infty & 0 & 0 & \infty & 3800 \\ 400 & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8500 & \infty & \infty & 600 & 0 & \infty \\ 3900 & \infty & 600 & \infty & 0 & \infty \\ 0 & \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & 0 & 4500 & 3800 & \infty \end{bmatrix}$$

$$\hat{c}(10) = 77000 + 7800 + 0 = 84800$$

Pohon ruang status yang terbentuk dari proses reduksi di atas terlihat pada Gambar 2.28.



Gambar 2.28 Pohon Ruang Status Level 2

Selanjutnya mengekspansi kembali titik dengan bobot minimum, dalam hal ini adalah titik 9.

10. Titik 11; lintasan 1,2,5,3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & 600 & \infty & 0 \\ 2900 & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & 0 & 0 & \infty & 3800 \\ 0 & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 600 & \infty & 0 \\ 2900 & \infty & \infty & \infty & \infty & 3900 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 600 & \infty & 0 \\ 0 & \infty & \infty & \infty & \infty & 1000 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 0 \\ 0 & \infty & \infty & \infty & \infty & 1000 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & 3900 & \infty & \infty \end{bmatrix}$$

Matriks diatas diperoleh dari pengurangan baris ke-4 oleh 2900 dan kolom ke-4 oleh 600. $\hat{c}(11) = 78000 + 0 + 2900 + 600 = 81500$.

11. Titik 12; lintasan 1,2,5,4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & 600 & \infty & 0 \\ 2900 & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & 0 & 0 & \infty & 3800 \\ 0 & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty & \infty & \infty \end{bmatrix}$$

$$\hat{c}(12) = 78000 + 0 + 0 = 78000$$

12. Titik 13; lintasan 1,2,5,6

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & 600 & \infty & 0 \\ 2900 & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & 0 & 0 & \infty & 3800 \\ 0 & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

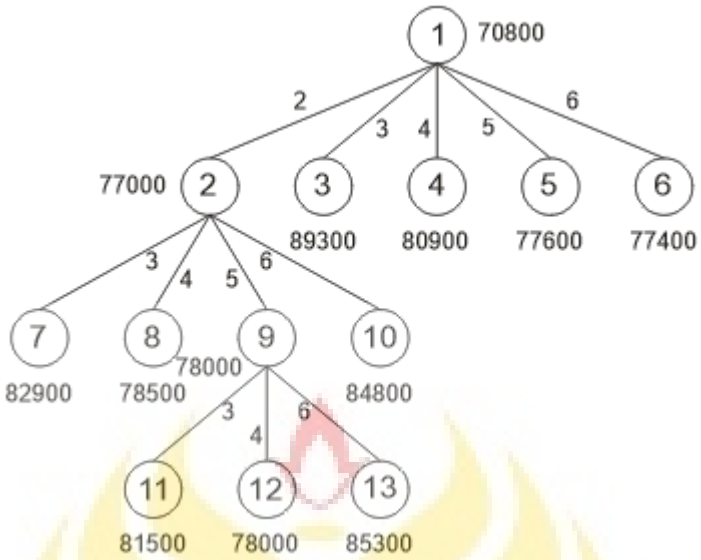
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & 600 & \infty & \infty \\ 2900 & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 7500 & \infty & \infty & 0 & \infty & \infty \\ 2900 & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & 0 & 4500 & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 4600 & \infty & \infty & 0 & \infty & \infty \\ 0 & \infty & 0 & \infty & \infty & \infty \\ \infty & \infty & 0 & 0 & \infty & \infty \\ \infty & \infty & 0 & 4500 & \infty & \infty \end{bmatrix}$$

Matriks di atas diperoleh dari pengurangan baris ke-3 oleh 600 dan kolom ke-1 oleh 2900. $\hat{c}(13) = 78000 + 3800 + 600 + 2900 = 85300$.

Pohon ruang status yang terbentuk dari proses reduksi di atas sampai saat ini terlihat pada Gambar 2.29.



Gambar 2.29 Pohon Ruang Level 3

Selanjutnya mengekspansi kembali titik dengan bobot minimum, dalam hal ini titik 12 dengan bobot 78000.

13. Titik 14; lintasan 1,2,5,4,3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$\hat{c}(14) = 78000 + 0 + 0 = 78000$$

14. Titik 15; lintasan 1,2,5,4,6

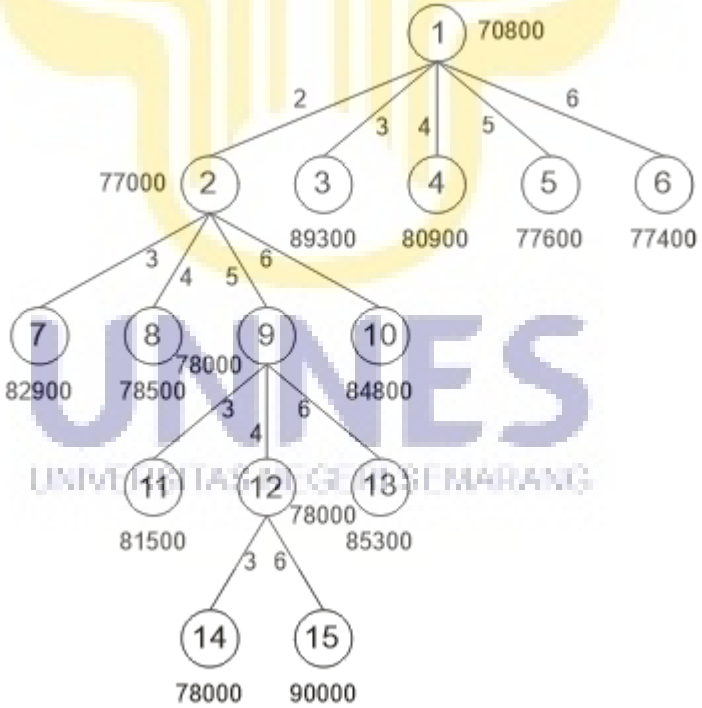
$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & 0 & \infty & \infty & 3900 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & 0 & \infty & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 8100 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \end{bmatrix} \approx$$

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & \infty \end{bmatrix} \approx$$

Matriks di atas diperoleh dari pengurangan baris ke-3 oleh 8100. $\hat{c}(15) = 78000 + 3900 + 8100 = 90000$.

Pohon ruang status yang terbentuk dari proses reduksi di atas sampai saat ini terlihat pada Gambar 2.30.



Gambar 2.30 Pohon Ruang Status Level 4

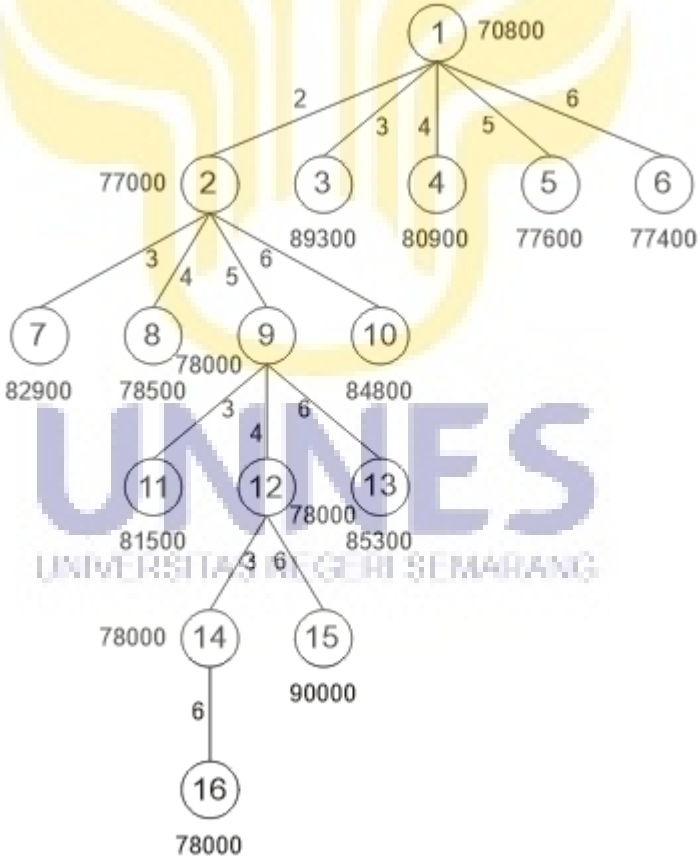
Selanjutnya mengekspansi kembali titik dengan bobot minimum, dalam hal ini titik 14 dengan bobot 7800.

15. Titik 16; lintasan 1,2,5,4,6

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty & \infty \end{bmatrix} \approx \begin{bmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

$$\hat{c}(16) = 78000 + 0 + 0 = 78000$$

Pohon ruang status yang terbentuk dari proses reduksi di atas sampai saat ini terlihat pada Gambar 2.31.



Gambar 2.31 Pohon Ruang Status level 5

Berdasarkan Gambar di atas dapat dijelaskan jika proses perhitungan dengan menggunakan algoritma *Branch and Bound* tersebut telah optimal. Berdasarkan keseluruhan proses reduksi yang telah dilakukan maka berhasil ditemukan sirkuit optimal, yaitu melalui titik 1-2-9-12-14-16 yang berarti melalui titik *Start*-T2-T5-T4-T3-T6-*Start*.

2.4. Algoritma Genetika

Algoritma genetika adalah suatu algoritma pencarian yang berbasis pada mekanisme seleksi alam dan genetika. Algoritma genetika merupakan salah satu algoritma yang sangat tepat digunakan dalam menyelesaikan masalah optimasi kompleks, yang sulit dilakukan oleh metode konvensional (Desiani, 2006: 187).

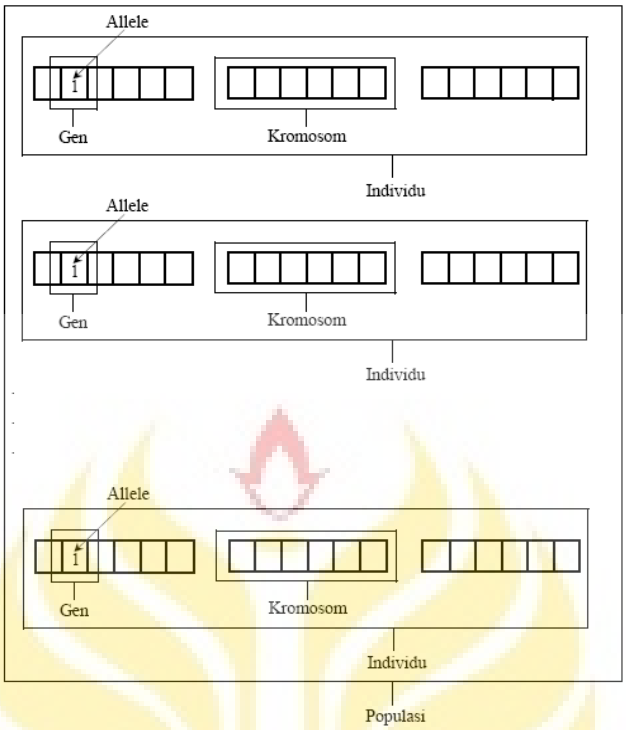
Menurut Philip, Taofiki & Kehinde (2011:26) algoritma genetika merupakan teknik yang digunakan untuk memperkirakan model komputer berdasarkan metode yang diadaptasi dari bidang genetika dalam ilmu biologi. Algoritma genetika merupakan salah satu metode penyelesaian optimasi yang dikenal mampu menghasilkan nilai optimum (Saptono & Hidayat, 2007).

Beberapa definisi penting yang perlu diperhatikan dalam membangun penyelesain masalah menggunakan algoritma genetika, yaitu sebagai berikut.

1. Genotip (Gen) adalah sebuah nilai yang menyatakan suatu dasar yang membentuk suatu arti tertentu dalam satu kesatuan gen yang dinamakan kromosom. Dalam algoritma genetika, gen ini bisa bernilai biner, *float*, integer maupun karakter.
2. Allel adalah nilai dari gen.
3. Kromosom adalah gabungan gen-gen yang membentuk nilai tertentu.

4. Individu merupakan satu 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.
6. Generasi menyatakan satu satunya siklus proses evolusi.
7. Nilai *fitness* menyatakan seberapa baik nilai dari individu atau solusi yang didapatkan.
8. *Phenotype* merupakan string (kromosom) yang merupakan solusi akhir.
9. *Genotype* merupakan sejumlah kromosom hasil perkawinan yang berpotensi sebagai solusi.
10. Locus merupakan letak suatu gen berada dalam suatu kromosom
11. *Offspring* merupakan anak (generasi berikutnya) yang terbentuk dari gabungan 2 kromosom generasi sekarang yang bertindak sebagai induk *(parent)* dengan menggunakan operator penyilangan *(crossover)* maupun operator mutasi (Basuki, 2003(b):3).

Gambar 2.32. berikut ini menjelaskan hubungan dari istilah-istilah yang didefinisikan di atas.



Gambar 2.32. Ilustrasi Definisi Individu Dalam Algoritma Genetika

Ciri-ciri permasalahan yang dikerjakan dengan menggunakan algoritma genetika adalah sebagai berikut.

1. Mempunyai fungsi tujuan optimalisasi non linear dengan banyak kendala yang juga non linear.
2. Mempunyai kemungkinan solusi yang jumlahnya tak berhingga.
3. Membutuhkan solusi "*real-time*" dalam arti solusi bisa didapat dengan cepat sehingga dapat diimplementasikan untuk permasalahan yang mempunyai perubahan yang cepat seperti optimasi pada pembebanan kanal pada komunikasi seluler.
4. Mempunyai multi-objektif dan multi-kriteria, sehingga diperlukan solusi yang dapat secara bijak diterima oleh semua pihak (Basuki, 2003a:4).

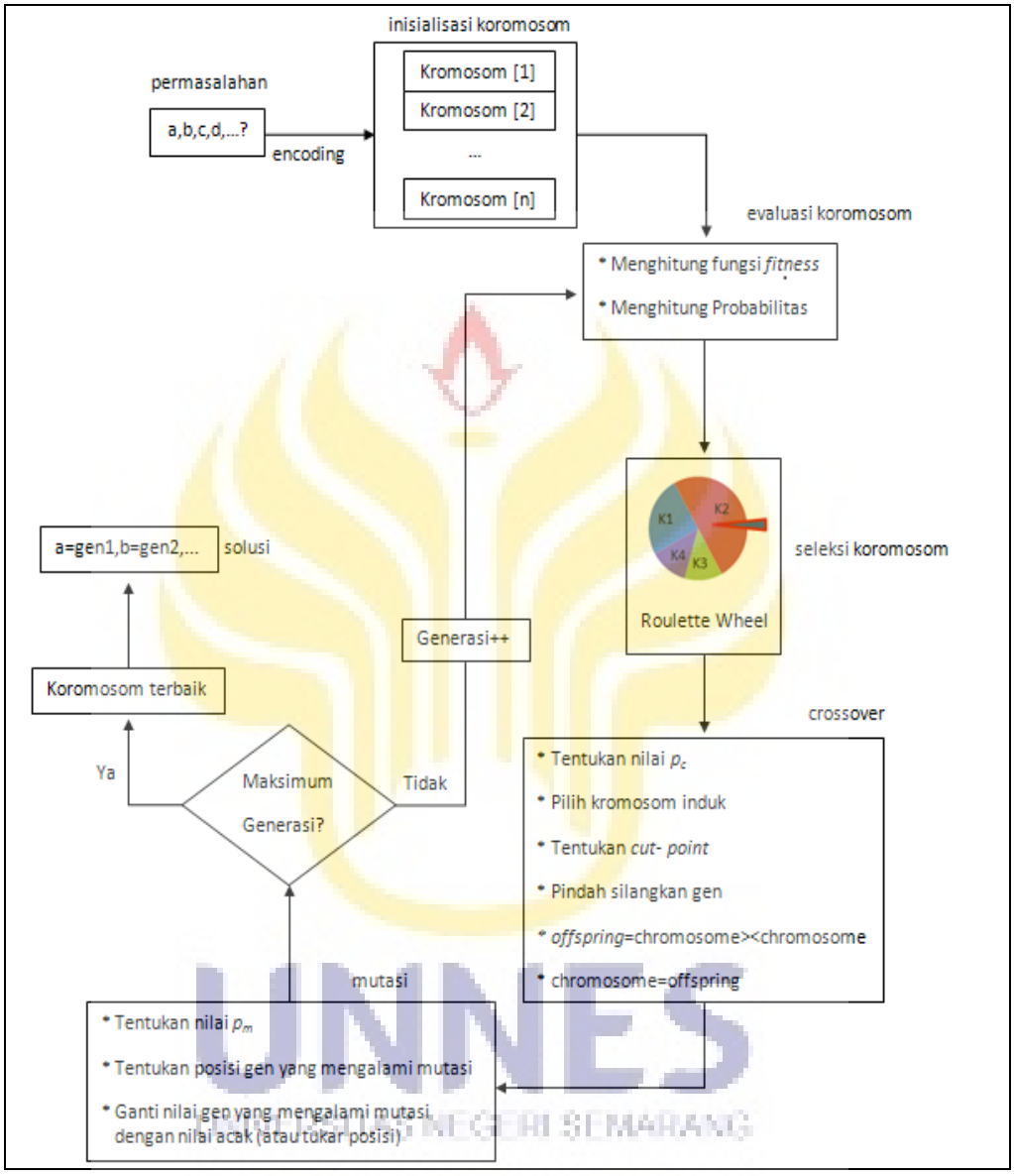
2.4.1. Struktur Umum Algoritma Genetika

Sebuah solusi yang dibangkitkan dalam algoritma genetika disebut sebagai kromosom, sedangkan kumpulan kromosom tersebut disebut sebagai populasi. Sebuah kromosom dibentuk dari komponen-komponen penyusun yang disebut sebagai gen dan nilainya dapat berupa bilangan numerik, biner, simbol ataupun karakter tergantung dari permasalahan yang ingin diselesaikan. Gen adalah komponen solusi dari suatu masalah yang akan diselesaikan. Kromosom-kromosom akan berevolusi secara berkelanjutan yang disebut dengan generasi. Pada setiap generasi kromosom-kromosom dievaluasi tingkat keberhasilan nilai solusinya terhadap masalah yang ingin diselesaikan menggunakan ukuran kebugaran yang disebut dengan *fitness* (istilah di dalam teknik optimasi, ini lebih dikenal sebagai fungsi tujuan -*objective function*- atau fungsi biaya -*cost function*-). *Fitness* sebagai masalah yang akan dioptimalkan. Jika nilai *fitness* semakim besar, maka sistem yang dihasilkan akan semakin baik. Fungsi *fitness* ini ditentukan dengan menggunakan metode heuristik.

Untuk memilih kromosom yang tetap dipertahankan untuk generasi selanjutnya dilakukan proses yang disebut dengan seleksi (dengan proses tertentu, misalnya teknik *roulette wheel* untuk memilih pasangan dari induk yang akan dikawinkan). Proses seleksi kromosom menggunakan konsep aturan evolusi Darwin yang telah disebutkan sebelumnya yaitu kromosom yang mempunyai nilai *fitness* tinggi akan memiliki peluang lebih besar untuk terpilih lagi pada generasi selanjutnya.

Dalam perkawinan tersebut akan terjadi proses pertukaran gen antar individu yang kawin tersebut. Ada dua operator utama dalam proses tersebut, yakni perkawinan silang (*crossover*) dan mutasi. Dari hasil perkawinan tersebut dihasilkan keturunan (anak) berupa kromosom baru. Kromosom-kromosom baru tersebut disebut dengan *offspring* (anak) yang membawa beberapa sifat dari induknya. Jumlah kromosom dalam populasi yang mengalami pindah silang ditetukan oleh paramater yang disebut dengan probabilitas pindah silang (*crossover probability, Pc*).

Mekanisme perubahan susunan unsur penyusun makhluk hidup akibat adanya faktor alam yang disebut dengan mutasi direpresentasikan sebagai proses berubahnya satu atau lebih nilai gen dalam kromosom dengan suatu nilai acak. Jumlah gen dalam populasi yang mengalami mutasi ditentukan oleh parameter yang dinamakan probabilitas mutasi (*mutation probability, Pm*). Setelah beberapa generasi akan dihasilkan kromosom-kromosom yang nilai gen-gennya konvergen ke suatu nilai tertentu yang merupakan solusi optimum (atau yang lebih dikenal sebagai *acceptable optimum*) yang dihasilkan oleh algoritma genetika terhadap permasalahan yang ingin diselesaikan. Struktur umum algoritma genetika digambarkan pada Gambar 2.33.



Gambar 2.33 Diagram Alir Algoritma Genetika

2.4.2. Komponen-Komponen Algoritma Genetika

Secara umum sebuah penerapan algoritma genetika akan melalui siklus sederhana yang terdiri dari 4 langkah, yaitu sebagai berikut (Zulfikar, 2008:13).

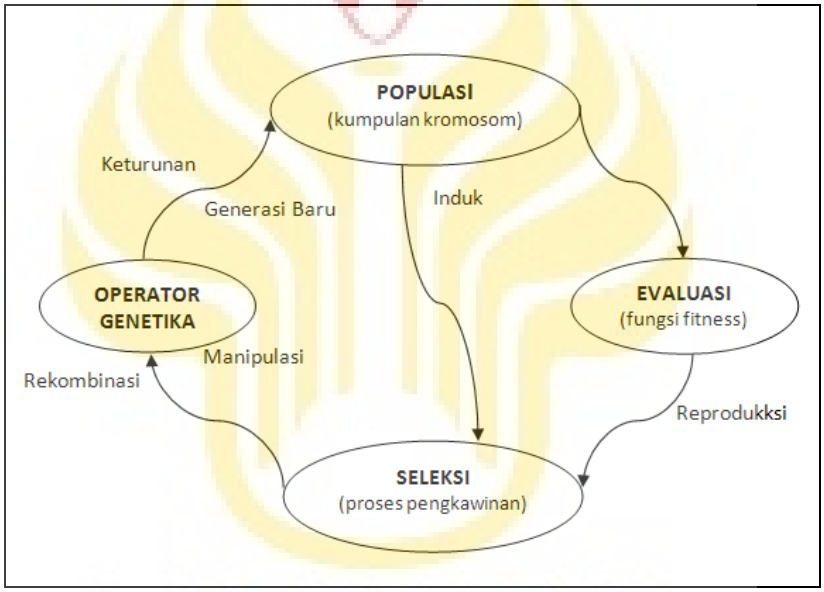
1. Membangun sebuah populasi yang terdiri dari beberapa string (kromosom).

2. Evaluasi masing-masing string (*fitness value*).

3. Proses seleksi agar didapat string yang terbaik.

4. Manipulasi genetika untuk menciptakan populasi baru dari string.

Ilustrasi mengenai siklus 4 langkah yang diinspirasi dari proses biologi untuk algoritma genetika dapat dilihat pada Gambar 2.34.



Gambar 2.34 Siklus Algoritma Genetika

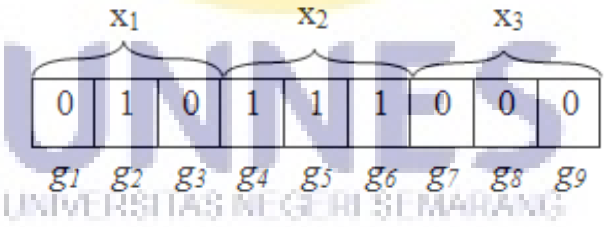
Setiap siklus yang dilalui memunculkan generasi baru yang dimungkinkan sebagai solusi bagi permasalahan yang ada. Pada dasarnya Algoritma Genetika memiliki tujuh komponen. Tetapi banyak metode yang bervariasi yang diusulkan pada masing-masing komponen tersebut. Masing-masing metode memiliki kelebihan dan kekurangan. Suatu metode yang bagus untuk penyelesaian masalah A belum tentu bagus untuk masalah B, atau bahkan tidak bisa digunakan untuk masalah C.

2.4.2.1. Skema Pengkodean

Pengkodean suatu kromosom adalah langkah pertama ketika kita menggunakan algoritma genetika untuk menyelesaikan suatu masalah. Pengkodean ini biasanya tergantung kepada masalah yang dihadapi. Pengkodean meliputi pengkodean terhadap gen yang terdapat dalam kromosom.

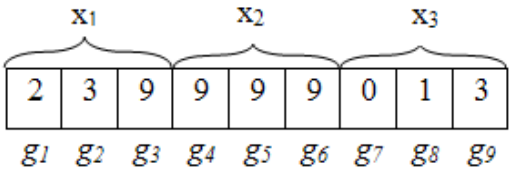
Gen merupakan bagian dari kromosom. Satu gen bisa mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk *string bit*, pohon, *array* bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika (Kusumadewi, 2003:14). Terdapat tiga skema yang paling umum digunakan dalam pengkodean, yaitu sebagai berikut.

(a) *Binary encoding*. Setiap gen hanya dapat bernilai 0 atau 1. Skema Pengkodean *Binary Encoding* terlihat pada Gambar 2.35.



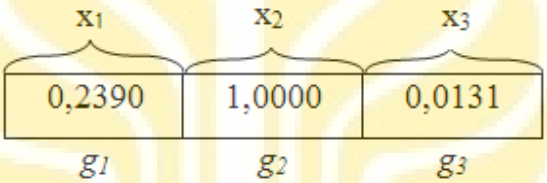
Gambar 2.35 Skema Pengkodean *Binary Encoding*

(b) *Discrete decimal encoding*. Setiap gen dapat bernilai salah satu bilangan bulat dalam interval [0,9]. Skema pengkodean *discrete decimal encoding* terlihat pada Gambar 2.36.



Gambar 2.36 Skema Pengkodean *Discrete Decimal Encoding*

(c) *Real-number encoding*. Pada skema ini, nilai gen berada dalam interval [0,R], di mana R adalah bilangan real positif dan biasanya R=1. Skema pengkodean *real-number encoding* terlihat pada Gambar 2.37.



Gambar 2.37 Skema Pengkodean *Real-Number Encoding*

(d) *Permutation Encoding*, setiap kromosom merupakan string dari sejumlah angka atau nomor yang merepresentasikan suatu posisi dalam suatu urutan.

Koromosom A 1 2 3 4 6 7 5 9 8

Koromosom B 2 4 6 7 8 9 1 5 3

Pada Gambar 2.35, 2.36 dan 2.37 skema pengkodean di atas terdapat tiga variabel, yaitu x_1, x_2, x_3 yang dikodekan ke dalam sebuah kromosom yang terdiri dari 9 gen (untuk *binary encoding* dan *discrete decimal encoding*) di mana setiap variabel dikodekan ke dalam 3 gen. Sedangkan pada *real-number encoding* ketiga variabel dikodekan ke dalam koromosom yang terdiri dari 3 gen, di mana masing-masing variabel dikodekan ke dalam 1 gen.

2.4.2.2. Nilai *Fitness*

Suatu individu dievaluasi berdasarkan suatu fungsi tertentu sebagai ukuran performansinya. Di dalam evolusi alam, individu yang memiliki nilai *fitness* tinggi yang akan bertahan hidup. Sedangkan individu yang bernilai *fitness* rendah akan mati. Langkah evaluasi *fitness* yaitu string dikonversi ke parameter fungsi, fungsi objektifnya dievaluasi, kemudian mengubah fungsi objektif tersebut ke dalam fungsi *fitness*.

Dalam masalah optimasi, jika masalah yang dicari adalah memaksimalkan sebuah fungsi *h* (dikenal sebagai masalah maksimasi) maka nilai *fitness* yang digunakan adalah nilai dari fungsi *h* tersebut, yakni *f = h* (di mana *f* adalah nilai *fitness*). Tetapi jika masalah adalah meminimalkan fungsi *h* (masalah minimasi), maka fungsi *h* tidak dapat digunakan secara langsung. Hal ini disebabkan adanya aturan bahwa individu yang memiliki *fitness* tertinggi lebih mampu bertahan hidup pada generasi berikutnya.

Untuk permasalahan minimalisasi, nilai *fitness* adalah inversi dari nilai maksimal yang diharapkan. Proses inversi dapat dilakukan dengan rumusan *fitness*$= \frac{1}{f(x)}$, di mana x merupakan kromosom (Basuki, 2003(a):17).

$$fitness = A - f(x) \text{ atau } fitness = \frac{A}{f(x)+\varepsilon}$$

Keterangan sebagai berikut.

A : konstanta yang ditentukan

x : individu (kromosom)

$f(x)$: nilai fungsi kromosom

ε : bilangan kecil yang ditentukan untuk menghindari pembagi nol atau $f(x) = 0$

2.4.2.3. Seleksi Induk

Menurut Khan, H. F. (1999), proses seleksi adalah proses mencari kromosom terbaik dalam satu generasi, di mana untuk menentukan suatu kromosom terbaik dapat dilihat dari nilai fitnessnya. Proses seleksi dilakukan dengan mengevaluasi setiap kromosom berdasarkan nilai fitnessnya untuk mendapatkan peringkat terbaik. Pada tahap ini, kromosom diseleksi sesuai dengan nilai fitnessnya. Tahap pertama yang dilakukan adalah nilai fitness yang diperoleh dijumlahkan, kemudian bangkitkan bilangan random. Setelah itu, nilai fitness yang telah terurut tadi dibandingkan dengan bilangan random yang dibangkitkan. Jika (nilai fitness)/(total fitness)> bilangan random yang telah dibangkitkan, maka kromosom tersebut akan terpilih sebagai induk untuk melakukan proses selanjutnya Beberapa metode yang biasa digunakan adalah sebagai berikut.

(a) *Rank-Based Fitness*

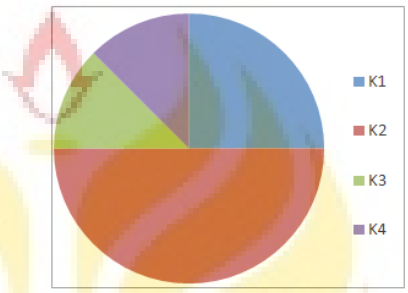
Pada *Rank-Based Fitness*, populasi diurutkan menurut nilai objektifnya. Nilai *fitness* tiap-tiap individu hanya tergantung pada posisi individu tersebut dalam urutan dan tidak dipengaruhi oleh nilai objektifnya.

(b) *Roulette-wheel*

Sesuai dengan namanya, metode ini menirukan permainan *roulette-wheel* di mana masing-masing kromosom menempati potongan lingkaran pada roda *roulette* secara proporsional sesuai dengan nilai *fitness*-nya. Metode seleksi dengan mesin *roulette* ini merupakan metode yang paling sederhana dan sering

dikenal dengan nama *stochastic sampling with replacement.* Kromosom yang memiliki nilai *fitness* lebih besar menempati potongan lingkaran yang lebih besar dibandingkan dengan kromosom yang bernilai *fitness* rendah. Gambar 2.38 mengilustrasikan sebuah contoh penggunaan *roulette wheel.*

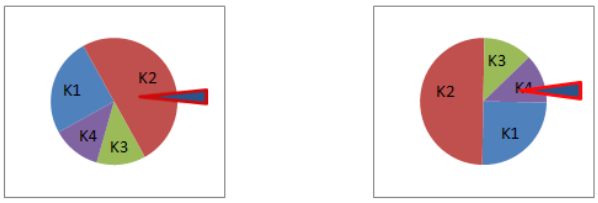
Kromosom	Nilai *Fitness*
K1	1
K2	2
K3	0.5
K4	0.5
Jumlah	4



Gambar 2.38 Contoh Penggunaan Metode *Roulette-Wheel*

Metode *roulette-wheel* sangat mudah diimplementasikan dalam pemrograman. Pertama, dibuat interval nilai kumulatif (dalam interval [0,1]) dari nilai *fitness* masing-masing kromosom dibagi total nilai *fitness* dari semua kromosom. Sebuah kromosom akan terpilih jika bilangan random yang dibangkitkan berada pada interval nilai kumulatifnya. Pada Gambar 2.38, K1 menempati interval nilai kumulatif [0;0,25], K2 berada dalam interval (0,25;0,75], K3 dalam interval (0,75;0,875] dan K4 dalam interval (0,875;1].

Misalkan, jika bilangan random yang dibangkitkan adalah 0,6 maka kromosom K2 terpilih sebagai orang tua. Tetapi jika bilangan random yang dibangkitkan 0,99 maka kromosom K4 yang terpilih. Seperti diilustrasikan pada Gambar 2.39.



Bilangan random = 0,6 Bilangan random=0,99

Gambar 2.39 Ilustrasi Prinsip Kerja Metode *Roulette-Wheel*

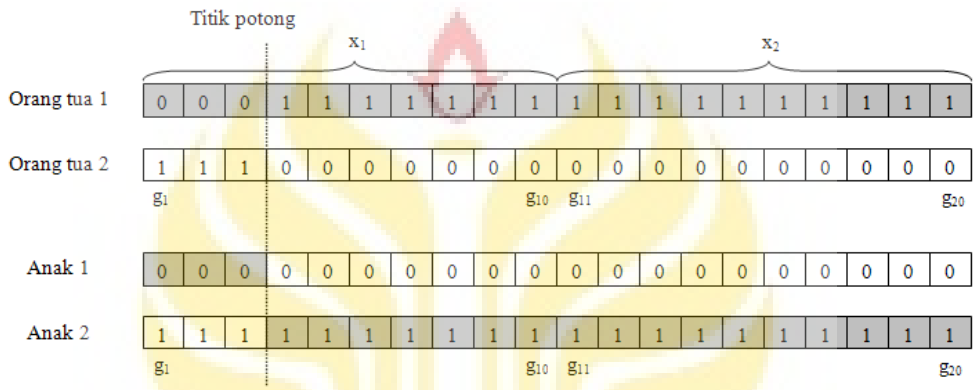
(c) *Tournament*

Pada seleksi alam yang terjadi di dunia nyata, beberapa individu (biasanya individu jantan) berkompetisi dalam sebuah kelompok kecil sampai tersisa hanya satu individu pemenang. Individu pemenang inilah yang bisa kawin (pindah silang). Metode *roulette-wheel selection* tidak mengkombinasikan masalah ini. Sebuah metode *tournament selection* mencoba mengadopsi karakteristik alami ini. Dalam bentuk paling sederhana, metode ini mengambil dua kromosom secara random dan kemudian menyeleksi salah satu yang bernilai *fitness* paling tinggi untuk menjadi orang tua pertama. Cara yang sama dilakukan lagi untuk mendapatkan orang tua yang kedua.

Metode *tournament* yang lebih rumit adalah dengan mengambil *m* kromosom secara random. Kemudian kromosom bernilai fitness tertinggi dipilih menjadi orang tua pertama, apabila bilangan random yang dibangkitkan kurang dari suatu nilai batas yang ditentukan *p* dalam interval [0,1). Pemilihan orang tua akan dilakukan secara random dari *m* - 1 koromosom yang ada jika bilangan yang dibangkitkan lebih dari atau sama dengan *p*. Pada *tournament selection*, variabel *m* adalah *tournament size* (ukuran grup) dan *p* adalah *tournament probability*. Biasanya *m* diset sebagai suatu nilai yang sangat kecil, misal 4 atau 5. Sedangkan *p* biasanya diset sekitar 0,75.

2.4.2.4. Pindah Silang *(Crossover)*

Salah satu komponen paling penting dalam algoritma genetika adalah *crossover* atau pindah silang. Sebuah kromosom yang mengarah pada solusi yakni bagus bisa diperoleh dari proses memindah-silangkan dua buah kromosom. Contoh pindah silang terlihat pada Gambar 2.40.



Gambar 2.40 Contoh Proses Pindah Silang

Pada Gambar 2.40 apabila solusi yang dicari adalah $x_1 = 0$ dan $x_2 = 0$, maka kromosom anak 1 memiliki nilai *fitness* tinggi dan menuju pada solusi yang dicari. Pindah silang bisa juga berakibat buruk jika ukuran populasinya sangat kecil. Dalam suatu populasi yang sangat kecil, suatu kromosom dengan gen-gen yang mengarah ke solusi akan sangat cepat menyebar ke kromosom-kromosom lainnya. Untuk mengatasi masalah ini digunakan suatu aturan bahwa pindah silang hanya bisa dilakukan dengan suatu probabilitas tertentu *Pc*. Artinya, pindah silang bisa dilakukan hanya jika suatu bilangan random [0,1] yang dibangkitkan kurang dari *Pc* yang ditentukan. Pada umumnya *Pc* diset mendekati 1, misalnya 0,8.

Pindah silang bisa dilakukan dalam beberapa cara berbeda, yang paling sederhana adalah pindah silang satu titik potong (*one-point crossover*). Suatu titik potong dipilih secara random, kemudian bagian pertama dari orang tua 1 digabung dengan bagian kedua dari orang tua 2. Untuk kromosom yang sangat panjang, misalkan 1000 gen, mungkin saja diperlukan beberapa titik potong. Pindah silang lebih dari satu titik potong disebut *n-point crossover*, di mana *n* titik potong dipilih secara random dan bagian-bagian kromosom dipilih dengan probabilitas 0,5 dari salah satu orang tuanya.

Crossover (perkawinan silang) bertujuan menambah keanekaragaman string dalam suatu populasi. Beberapa jenis *crossover* tersebut adalah sebagai berikut (Kusumadewi, 2003:14).

(a) *Crossover Diskret*

Proses *crossover* dilakukan dengan menukar nilai variabel antar kromosom induk. Misalkan ada 2 individu dengan 3 variabel, yaitu sebagai berikut.

Induk 1: 12 25 5

Induk 2: 123 4 34

untuk tiap-tiap variabel induk yang menyumbang variabelnya ke anak dipilih secara acak dengan probabilitas yang sama.

Sampel 1: 2 2 1

Sampel 2: 1 2 1

Kromosom baru yang terbentuk:

Anak 1: 123 4 5

Anak 2: 12 4 5

(b) *Crossover Intermediate* (menengah)

Crossover menengah merupakan metod *crossover* yang hanya dapat digunakan untuk variabel *real*. Nilai variabel anak dipilih di sekitar dan antara nilai-nilai variabel induk. Anak dihasilkan menurut aturan sebagai berikut.

Anak = induk 1 + alpha (induk 2 – induk 1)

Dengan alpha adalah faktor skala yang dipilih secara random pada interval [-d, 1+d], biasanya d=0,25. Tiap-tiap variabel pada anak merupakan hasil *crossover* variabel-variabel menurut aturan di atas dengan nilai alpha dipilih ulang untuk tiap variabel.

Misalkan ada 2 individu dengan 3 variabel, yaitu sebagai berikut.

Induk 1: 12 25 5

Induk 2: 123 4 34

Misalkan nilai alpha yang terpilih adalah sebagai berikut.

Sampel 1: 0,5 1,1 -0,1

Sampel 2: 0,1 0,8 0,5

Kromosom baru yang terbentuk

Anak 1: 67,5 1,9 2,1

Anak 2: 23,1 8,2 19,5

(c) *Crossover* Garis

Pada dasarnya *crossover* garis ini sama dengan *crossover* menengah, hanya saja nilai alha untuk semua variabel sama. Misalkan ada 2 individu dengan 3 variabel, yaitu sebagai berikut.

Induk 1: 12 25 5

Induk 2: 123 4 34

Alpha yang dipilih adalah sebagai berikut.

Sampel 1: 0,5

Sampel 2: 0,1

Kromosom baru yang terbentuk adalah sebagai berikut.

Anak 1: 67,5 14,5 19,5

Anak 2: 23,1 22,9 7,9

(d) *Crossover* dengan permutasi

Pada penyilangan permutasi ini kromosom-kromosom anak diperoleh dengan cara memilih sub barisan tour dari satu induk dengan tetap menjaga urutan dan posisi sejumlah gen yang mungkin terhadap induk yang lainnya.

Contoh *crossover* dengan permutasi sebagai berikut.

Misal

Induk 1: (1 2 3 | 4 5 6 7 | 8 9)

Induk 2: (4 5 3 | 1 8 7 6 | 9 2)

Anak 1: (x x x | 1 8 7 6 | x x)

Anak 2: (x x x | 4 5 6 7 | x x)

Dari sini kita memperoleh pemetaan sebagai berikut.

1-4, 8-5, 7-6, 6-7

Kemudian *copy* sisa gen di induk ke anak 1 dengan menggunakan pemetaan yang sudah ada.

Anak 1: (1-4 2 3 | 1 8 7 6 | 8-5 9)

Anak 1: (4 2 3 | 1 8 7 6 | 5 9)

Lakukan hal yang sama untuk anak 2.

Anak 2: (1-4 5-8 3 | 1 8 7 6 | 9 2)

(e) Order Cossover

Order crossover merupakan cara *crossover* dengan menukar kromosom dengan tetap menjaga urutan gen yang bukan bagian dari kromosom tersebut.

Contoh *order crossover* adalah sebagai berikut.

Misalkan ada 3 kromosom induk yang akan dilakukan *crossover* sebagai berikut.

Kromosom [1] = [ABCD]

Kromosom [2] = [BACD]

Kromosom [3] = [ACDB]

Proses *crossover* sebagai berikut.

Kromosom [1] = Kromosom [1] >< Kromosom [2]

= [ABCD] >< [BACD]

= [ACBD]

Kromosom [2] = Kromosom [2] >< Kromosom [3]

= [BACD] >< [BCDA]

= [BCDA]

Kromosom [3] = Kromosom [3] >< Kromosom [1]

= [BCDA] >< [ABCD]

= [BCAD]

2.4.2.5. Mutasi

Prosedur mutasi sangatlah sederhana. Untuk semua gen yang ada, jika bilangan random yang dibangkitkan kurang dari probabilitas mutasi *Pm* yang ditentukan maka ubah gen tersebut menjadi nilai kebalikannya (dalam binary encoding, 0 diubah 1, dan 1 diubah 0). Dalam *swapping mutation* dilakukan pertukaran tempat node. Biasanya *Pm* diset sebagai 1/*n*, di mana *n* adalah jumlah gen dalam kromosom. Dengan *Pm* sebesar ini berarti mutasi terjadi pada sekitar satu gen saja. Pada algoritma genetika sederhana nilai *Pm* tetap selama evolusi.

Mutasi merupakan proses mengubah nilai satu atau beberapa gen dalam satu kromosom. Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi.

(a) Mutasi dengan Pengkodean Biner

Mutasi dengan pengkodean biner merupakan operasi yang sangat sederhana. Proses yang dilakukan adalah menginversi nilai *bit* pada posisi tertentu yang dipilih secara acak (dengan menggunakan skema tertentu) pada kromosom.

Contoh mutasi pada pengkodean biner sebagai berikut.

Kromosom sebelum mutasi : 1 0 0 1 0 1 1 1

Kromosom sesudah mutasi : 1 0 0 1 0 0 1 1

(b) Mutasi dengan Pengkodean Permutasi

Proses mutasi yang dilakukan dalam pengkodean biner tidak dapat dilakukan pada pengkodean permutasi karena konsistensi urutan permutasi harus

diperhatikan. Salah satu cara yang dapat dilakukan adalah dengan memilih dua posisi (*locus*) dari kromosom dan kemudian nilainya saling dipertukarkan.

Contoh mutasi dalam pengkodean permutasi sebagai berikut.

Kromosom sebelum mutasi : 1 2 3 4 5 6 7 8 9

Kromosom sesudah mutasi : 1 2 7 4 6 5 8 3 9

(c) Mutasi dengan Pengkodean Nilai

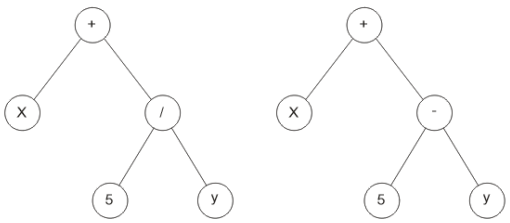
Proses mutasi dalam pengkodean nilai dapat dilakukan dengan berbagai cara, salah satunya yaitu dengan memilih sembarang posisi gen pada kromosom. Nilai yang ada tersebut kemudian ditambahkan atau dikurangkan dengan nilai kecil tertentu yang diambil secara acak. Contoh mutasi dalam pengkodean nilai real dengan nilai yang ditambahkan atau dikurangkan adalah 0,1.

Kromosom sebelum mutasi : 1,43 1,09 4,51 9,11 6,94

Kromosom sesudah mutasi : 1,43 1,19 4,51 9,01 6,94

(d) Mutasi dengan Pengkodean Pohon.

Mutasi dalam pengkodean pohon dapat dilakukan antara lain dengan cara mengubah operasi (+, -, *, /) atau nilai yang terkandung dalam suatu *vertex* pohon yang dipilih, atau dapat juga dilakukan dengan memilih dua *vertex* dari pohon dan saling mempertukarkan operator atau nilainya. Contoh mutasi dalam pengkodean pohon seperti pada Gambar 2.41.



Gambar 2.41 Gen Sebelum dan Setelah Mutasi dengan Pengkodean Pohon

(e) *Swapping mutation*

Proses mutasi dengan cara ini dilakukan dengan menetukan jumlah kromosom yang akan mengalami mutasi dalam satu populasi melalui parameter *mutation rate* (*pm*). Proses mutasi dilakukan dengan cara menukar gen yang telah dipilih secra acak dengan gen sesudahnya, jika gen tersebut berada di akhir kromosom, maka ditukar dengan gen yang pertama.

Pertama hitung panjang total gen yang ada pada suatu populasi.

$$Panjang\ total\ gen\ =\ gen\ dalam\ 1\ kromosom\ x\ jumlah\ kromosom$$

Untuk memilih posisi gen yang akan mengalami mutasi dilakukan dengan membangkitkan bilangan acak antara 1 sampai panjang total gen untuk dilakukan proses mutasi. (Wibowo, 2003:14)

2.4.2.6. Elitisme

Karena seleksi dilakukan secara random, maka tidak ada jaminan bahwa suatu individu bernilai *fitness* tertinggi akan selalu terpilih. Kalaupun individu bernilai *fitness* tertinggi terpilih, mungkin saja individu tersebut akan rusak (nilai *fitness*-nya menurun) karena proses pindah silang. Untuk menjaga agar individu bernilai *fitness* tertinggi tersebut tidak hilang selama evolusi, maka perlu dibuat satu atau beberapa kopinya. Prosedur ini dikenal sebagai *elitisme*.

2.4.2.7. Pergantian Populasi

Dalam algoritma genetika dikenal skema penggantian populasi yang disebut *generational replacement*, yang berarti semua individu (misal *N* individu dalam suatu populasi) dari suatu generasi digantikan sekaligus oleh *N* individu baru hasil pindah silang dan mutasi. Skema penggantian ini tidak realistis dari

sudut pandang biologi. Di dunia nyata, individu-individu dari generasi berbeda bisa berada pada waktu yang bersamaan. Fakta lainnya adalah individu-individu muncul dan hilang secara konstan, tidak pada generasi tertentu. Secara umum skema pergantian populasi dapat dirumuskan berdasarkan suatu ukuran yang disebut *generational gap G*. Ukuran ini meunjukan persentase populasi yang digantikan dalam setiap generasi. Pada skema *generational replacement*, $G = 1$.

Skema pergantian yang paling ekstrem adalah hanya mengganti satu individu dalam setiap generasi, yaitu $G = 1/N$, di mana N adalah jumlah individu dalam populasi. Skema pergantian ini disebut sebagai *Steady-state reproduction.* Pada skema tersebut, G biasanya sama dengan $1/N$ atau $2/N$. Dalam setiap generasi, sejumlah NG individu harus dihapus untuk menjaga ukuran populasi tetap N. Terdapat beberapa prosedur penghapusan individu, yaitu penghapusan individu yang bernilai fitness paling rendah atau penghapusan individu yang paling tua. Penghapusan bisa berlaku hanya pada individu orang tua saja atau bisa juga berlaku pada semua individu dalam populasi (Zulfikar, 2008:20).

2.4.3. Penentuan Parameter Algoritma

Parameter algoritma genetika merupakan salah satu bagian penting dalam penerapan algoritma genetika yang tidak mudah ditentukan secara pasti. Tidak ada aturan yang pasti untuk menentukan parameter algoritma, baik probabilitas *cossover,* probabilitas mutasi, maupun ukuran populasi. Hal ini tidak terlepas dari prinsip algoritma genetika yang mengandalkan bilangan acak hampir dalam setiap langkahnya, mulai dari pembentukan populasi awal, proses *crossover*, atau proses

mutasi. Bilangan acak yang berbeda pasti akan menyebabkan hasil yang berbeda pula (Zukhri, 2014:49).

Parameter algoritma yang disarankan Hopgood, sebagaimana dikutip oleh Zukhri (2014:49) adalah sebagai berikut.

a. Probabilitas *crossover* cukup besar (berkisar 60% sampai 70%).
b. Probabilitas mutasi cukup kecil (sebuah gen untuk sebuah kromosom).
c. Ukuran populasi berkisar antara 50 sampai 500 kromosom.

2.5. *Travelling Salesman Problem* dalam Algoritma Genetika

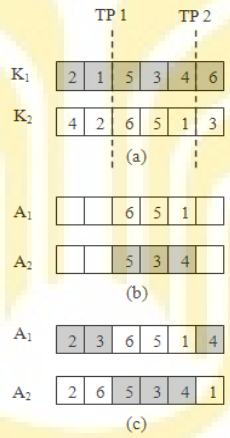
TSP dapat dirumuskan sebagai berikut: terdapat sekumpulan *N* node dengan posisi-posisi koordinatnya {*x*,*y*}, *i* = 1, 2, 3, …, *N*. Algoritma genetika dapat digunakan untuk menyelesaikan masalah TSP ini dengan cara sebagai berikut.

Suatu solusi direpresentasikan ke dalam suatu koromosom yang berisi nomor urut dari semua node yang ada. Masing-masing nomor urut node hanya boleh muncul satu kali di dalam koromosom sehingga satu koromosom merepresentasikan satu sirkuit atau satu solusi yang valid. Representasi ini disebut sebagai *permutation encoding*, di mana suatu kromosom merepresentasikan suatu permutasi dari nomor urut kota 1, 2, 3, …,*N*.

Kromosom-koromosom tersebut diperiksa nilai *fitness*-nya. Kromosom yang nilai *fitness*-nya paling tinggi akan dipilih untuk terus hidup pada generasi berikutnya dan berpeluang melakukan *crossover* untuk menghasilkan kromosom atau individu baru yang diharapkan mempunyai nilai *fitness* yang lebih baik. Dengan adanya mutasi diharapkan dapat memperbaiki kromosom yang sudah ada.

Individu-individu pada generasi-generasi berikutnya diharapkan akan memiliki nilai *fitness* yang lebih baik dan mengarah pada suatu solusi yang diharapkan. Solusi yang diambil adalah solusi pada individu atau kromosom yang paling besar nilai *fitness*-nya.

Dalam TSP, pindah silang dapat diimplementasikan dengan skema *order crossover*. Pada skema ini satu bagian koromosom dipertukarkan dengan tetap menjaga urutan node (gen) yang bukan bagian dari koromosom tersebut. ilustrasi skema *order crossover* terlihat pada Gambar 2.42.



Gambar 2.42 Pindah Silang Menggunakan *Order Crossover*

Mula-mula dua buah titik potong TP1 dan TP2, dibangkitkan secara random untuk memotong dua buah kromosom induk (orang tua), K1 dan K2 (Gambar 2.42.a). Kemudian dua koromosom anak, A1 dan A2 mendapat gen-gen dari bagian koromosom K1 dan K2 secara menyilang. Koromosom A1 mendapatkan {6, 5, 1} dan A2 mendapatkan {5, 3, 4} (Gambar2.42.b). Posisi gen yang masih kosong pada A1 diisi dengan gen-gen dari K1, secara berurutan dari gen 1 sampai gen 6, yang belum ada pada A1. Hal yang sama juga dilakukan untuk koromosom A2 (Gambar2.42.c).

Pada TSP, operator mutasi biasanya diimplementasikan dengan menurunkan gen termutasi dengan gen lain yang terpilih secara random. Misalnya, kromosom {2, 3, 4, 5, 6, 1} dapat termutasi menjadi koromosom {2, 6, 4, 5,3,1}. Dalam hal ini gen 3 dan 6 saling ditukar. Skema mutasi ini dikenal dengan *swapping mutation*.

2.6. *Matrix Laboratory*

Matlab merupakan bahasa pemrograman yang hadir dengan fungsi dan karakteristik yang berbeda dengan bahasa pemrograman lain yang sudah ada lebih dahulu seperti Delphi, Basic maupun C++. Matlab merupakan bahasa pemrograman level tinggi yang dikhususkan untuk kebutuhan komputasi teknis, visualisasi dan pemrograman seperti komputasi matematik, analisis data, pengembangan algoritma, simulasi dan pemodelan dan grafik-grafik perhitungan.

Matlab adalah sebuah bahasa dengan (high-performance) kinerja tinggi untuk komputasi masalah teknik. Matlab mengintegrasikan komputasi, visualisasi, dan pemrograman dalam suatu model yang sangat mudah untuk pakai di mana masalah-masalah dan penyelesaiannya diekspresikan dalam notasi matematika yang familiar (Iqbal, 2009:2). *Software* Matlab memiliki *tools* yang dapat memudahkan dalam proses pembuatan program (Purnamasari, Dwijanto & Sugiharti, 2013).

Matlab hadir dengan membawa warna yang berbeda. Hal ini karena Matlab membawa keistimewaan dalam fungsi-fungsi matematika, fisika, statistik, dan visualisasi. Matlab dikembangkan oleh MathWorks, yang pada awalnya dibuat untuk memberikan kemudahan mengakses data matrik pada proyek Linpack dan

Eispack. Saat ini Matlab memiliki ratusan fungsi yang dapat digunakan sebagai problem solver mulai dari simple sampai masalah-masalah yang kompleks dari berbagai disiplin ilmu (Firmansyah, 2007).

2.6.1. Menjalankan Matlab

Berikut langkah-langkah yang dilakukan untuk menjalankan program Matlab.

1. Klik pada tombol Start.
2. Pilih *All Programs*.
3. Klik pada folder Matlab.
4. Klik pada ikon Matlab R2009a (atau yang sesuai pada computer kita).

2.6.2. Menggunakan Variabel

Pada *Command Window*, kita bisa menggunakan variabel. Variabel adalah suatu nama yang dapat dipakai untuk menyimpan suatu nilai dan nilai yag ada di dalamnya bisa diubah sewaktu-waktu. Sebelum mempraktikkan penggunaan variabel, aturan tentang cara menamakan variabel perlu diketahui terlebih dahulu. Aturan dalam memberikan nama variabel adalah sebagai berikut.

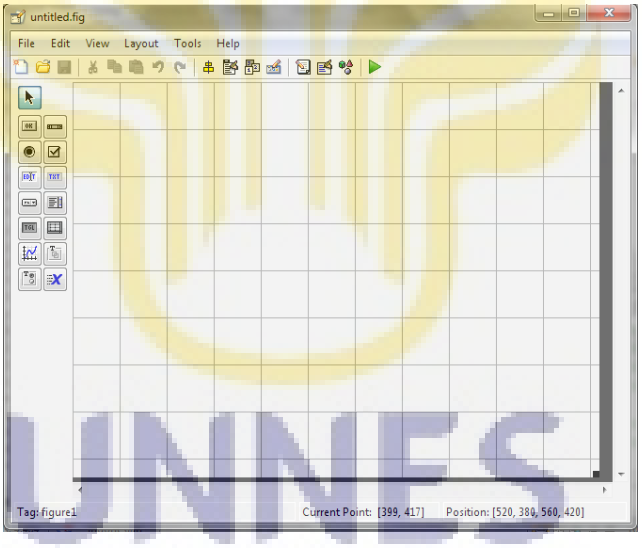
1. Matlab membedakan huruf kecil dan huruf kapital pada penamaan variabel. Dengan demikian bilangan dan Bilangan adalah dua variabel yang berbeda.
2. Nama variabel arus diawali dengan huruf sedangkan kelanjutannya dapat berupa huruf, angka atau tanda garis bawah (_).
3. Panjang nama variabel dapat mencapai 31 karakter. Jika nama variabel lebih dari 31 karakter, maka karakter ke-3 dan seterusnya diabaikan.

2.6.3. Mengenal GUI

GUI merupakan tampilan grafis yang memudahkan *user* berinteraksi dengan perintah teks. Dengan GUI, program yang dibuat menjadi lebih *user friendly*, Sehingga *user* mudah menjalankan suatu aplikasi program (Paulus & Nataliani, 2007:17).

Untuk membuka lembar kerja GUI dalam Matlab, kita menggunakan perintah `File - New- GUI` atau dengan mengetikkan `>> guide` pada Command Window.

Tampilan lembar kerja GUI dalam Matlab terlihat pada Gambar 2.43.



Gambar 2.43. Tampilan lembar Kerja GUI.

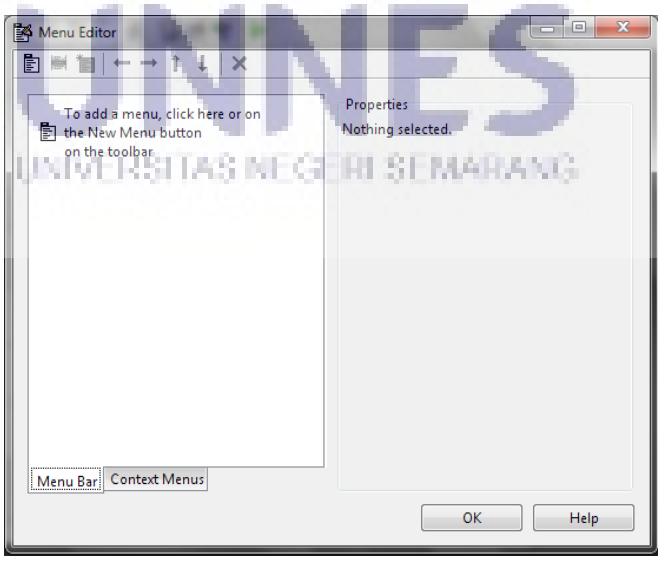
2.6.3.1. Toolbar GUI

Berikut adalah penjelasan kegunaan ikon-ikon pada toolbar GUI.

1. *New*  , untuk membuka lembar kerja GUI Matlab yang baru.
2. *Open*  , untuk membuka file Matlab yang sudah tersimpan.
3. *Save*  , untuk menyimpan GUI yang telah dibuat.

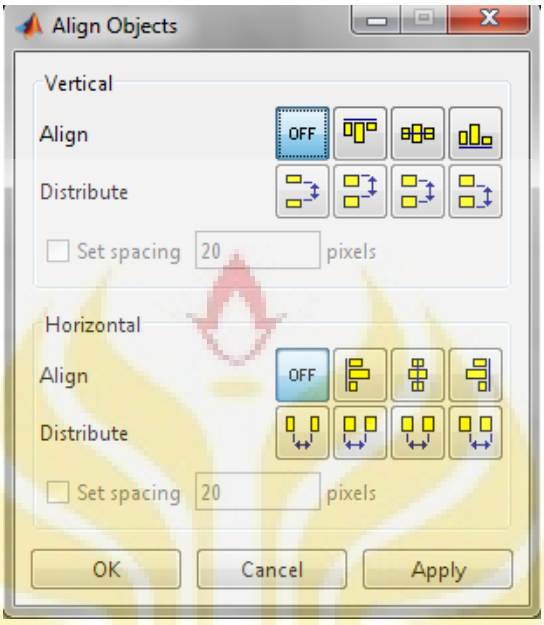
4. *Cut* , untuk menghapus komponen GUI supaya dapat disalin kembali.
5. *Copy* , untuk mengkopi komponen GUI supaya dapat disalin.
6. *Paste* , untuk menyalin komponen GUI yang telah dihapus atau dikopi.
7. *Undo* , untuk mengembalikan suatu perintah yang dilakukan sebelumnya.
8. *Redo* , untuk mengembalikan suatu perintah yang dilakukan sebelumnya.
9. *M-File Editor* , untuk membuka *script* program GUI pada *m-file editor*.
10. *Menu Editor* , terdapat dua menu,yaitu sebagai berikut.
    a. *Menu bar*: untuk membuat menu pada *figure* yang bersangkutan.
    b. *Context Menu*: akan tampil jika pengguna mengklik kanan mouse pada komponen di menu yang didefinisikan.

Tampilan *Menu Editor* terlihat pada Gambar 2.44.



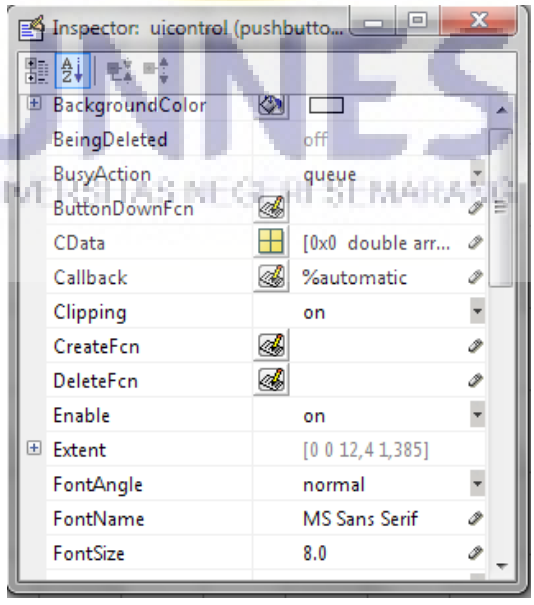
Gambar 2.44. Tampilan *Menu Editor*.

11. *Align Objects* , untuk merapikan beberapa komponen GUI. Tampilan *Align Object* terlihat pada Gambar 2.45



Gambar 2.45. Tampilan *Align Object*.

12. *Property Inspector* , untuk membuka properti suatu komponen GUI yang dibuat. Tampilan *Property Inspector* terlihat pada Gambar 2.46



Gambar 2.46. Tampilan *Property Inspector*.

13. *Object Browser* , untuk menampilkan daftar urutan komponen-komponen GUI pada *figure*.

14. *Run* , untuk menjalankan program.

2.6.3.2. Komponen- Komponen GUI

Komponen-komponen GUI terlihat pada Gambar 2.47.

Gambar 2.47. Komponen-komponen GUI.

1. *Push Button* , *push button* merupakan tombol yang jika diklik akan menghasilkan suatu tindakan.

2. *Slider* , *slider* menerima masukan berupa angka pada suatu *range* tertentu di mana pengguna menggeser kontrol pada *slider*.

3. *Radio Button* , *radio button* merupakan kontrol yang digunakan untuk memilih satu pilihan dari beberapa pilihan yang ditampilkan.

4. *Check Box* , *check box* merupakan kontrol yang digunakan untuk memilih satu atau lebih pilihan dari beberapa pilihan yang ditampilkan.

5. *Edit Text*  , *edit text* merupakan kontrol untuk memasukkan atau memodifikasi teks.

6. *Static Text*  , *static text* merupakan kontrol untuk membuat teks label.

7. *Pop Up Menu*  , *pop up menu* merupakan kontrol yang digunakan untuk membuka tampilan daftar daftar pilihan yang telah didefinisikan dengan mengklik tanda panah yang terdapat pada *pop up menu*.

8. *Listbox*  , *listbox* merupakan kontrol yang digunakan untuk menampilkan semua daftar item. Kemudian pengguna memilih satu diantara item-item yang ada.

9. *Toggle Button*  , *toggle button* hampir sama dengan *push button*, hanya jika *push button* diklik, tombol akan kembali ke posisi semula. Sebaliknya, jika *toggle button* diklik, tombol tidak akan kembali ke posisi semula kecuali diklik kembali.

10. *Axes*  , *axes* digunakan untuk menampilkan grafik atau gambar.

11. *Panel*  , *panel* merupakan kotak yag digunakan untuk menandai atau mengelompokkan daerah tertentu pada *figure*.

12. *Button group*  , *button group* hampir sama dengan *panel*, tetapi *button group* lebih digunakan untuk mengelompokkan *radio button* dan *toggle button.*

2.6.4. Merancang *Interface*

1. Menambahkan *Static Text.*

Berikut adalah langkah untuk menambah sebuah komponen *static text* ke dalam *interface* yang masih kosong.

(a) Klik pada *Static Text* pada *toolbar*.

(b) Letakkan penunjuk *mouse* pada posisi lalu di klik.

(c) Klik pada ikon *property inspector* maka akan muncul jendela *inspektor.*

(d) Letakkan penunjuk mouse pada posisi dikanan property bernama *string.*

(e) Kliklah lalu gantilah *static text* menjadi kata yang diinginkan, kemudian tekan tombol *enter*.

2. Menambahkan *Edit Text*.

Berikut adalah langkah untuk menambahkan sebuah komponen *Edit Text*.

(a) Klik pada *Edit Text* pada *toolbar*.

(b) Letakkan penunjuk *mouse* pada posisi yang diinginkan.

(c) Klik kotak *edit (Edit Text)* yang baru saja kita tambahkan perlu diberi nama. Cara memberi nama yaitu sebagai berikut.

a. Aktifkan *Property Inspektor*.

b. Gantikan isi *Property Tag* menjadi misal edit_...

c. Kosongkan isi *Property String*.

Dengan cara seperti itu, nama untuk kotak *edit* diberi nama edit_... dan isinya dikosongkan.

3. Menambah *Push Button.*

Berikut adalah langkah untuk menambah sebuah komponen *Push Button.*

(a) Klik pada *Push Button* pada *toolbar.*

(b) Letakkan penunjuk mouse pada posisi yang diinginkan.

(c) Tombol yang baru saja kita tambahkan perlu diberi nama dan dilengkapi dengan judul. Cara memberi nama dan judul sebagai berikut.

a. Aktifkan *Property Inspektor.*

b. Gantilah isi *Property Tag* menjadi pushbutton_hitung.

c. Tekan hitung pada *Property String* dan kemudian tekanlah tombol *enter.*

2.6.5. Menyimpan *Interface*

Setelah desain *Interface* dibuat. Kita perlu menyimpannya terlebih dahulu. Caranya seperti berikut.

1. Klik pada menu *File.*
2. Klik pada pilihan *Save.*
3. Ketikkan nama *File* pada *file name.*
4. Klik pada tombol *Save.*

Dengan cara seperti itu, judul pada *interface* berubah menjadi nama *file* yang kita simpan (...).fig. Setelah penyimpanan dilakukan, Matlab membentuk skrip dengan nama (...).m. Melalui skrip tersebut, penambahan kode secara manual akan dilakukan.

2.6.6. Menjalankan *Interface*

Untuk menguji *Interface* yang telah kita buat, kita bisa mengetikkan *file* yang akan diuji pada *Command Windows* dan menekan tombol enter. Cara yang lebih mudah klik saja pada ikon  yang terdapat pada *interface* yan kita buat.

BAB V

PENUTUP

5.1. Kesimpulan

Berdasarkan hasil penelitian dan pembahasan mengenai perbandingan algoritma *branch and bound* dan algoritma genetika untuk mengatasi *Travelling Salesman Problem* (TSP) menggunakan *software* Matlab, dapat diambil kesimpulan sebagai berikut.

1. Penerapan algoritma *branch and bound* dan algoritma genetika guna menentukan sirkuit terpendek dalam pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang dimulai dengan mencari jarak antar alamat dengan bantuan *Google Maps*, kemudian dilanjutkan dengan pembangunan sistem TSP yang dilengkapi dengan koding pada *software* Matlab R2009a. Setelah sistem TSP berhasil dibuat selanjutnya inputkan data alamat yang telah disimpan dalam *database*, *inputkan* pula variabel-variabel masukan seperti ukuran populasi, probabilitas *crossover*, probabilitas mutasi dan generasi. Selanjutnya dilakukan pengujian sistem dengan melakukan modifikasi pada algoritma genetika. Hasil modifikasi pada algoritma genetika yang paling optimal digunakan untuk perbandingan dengan hasil perhitungan algoritma *branch and bound.* Selanjutnya akan didapatkan hasil perbandingan algoritma *branch and bound* dan algoritma genetika yang mempunyai panjang sirkuit terpendek.

2. Berdasarkan solusi optimum yang diperoleh dengan menggunakan algoritma *branch and bound* Sebesar 67,9 Km dan dengan algoritma genetika diperoleh panjang sirkuit yang lebih pendek dari panjang sirkuit yang dihasilkan algoritma *branch and bound* yaitu sebesar 61.9 Km. Hal ini menunjukkan bahwa algoritma genetika lebih efektif dalam menentukan sirkuit terpendek untuk pengiriman barang di PT. Jalur Nugraha Ekakurir (JNE) Semarang. Sirkuit terpendek yang dihasilkan pada proses pendistribusian barang menggunakan algoritma genetika melalui JNE, Jalan Kyai Saleh No. 10- Tambakaji- Jalan Melati Baru 5- Jalan Penjaringan I- Bangetayu Wetan- Jalan Sedayu Indah- Jalan Kauman Timur- Jalan Lamper Tengah- Jalan Tirto Agung- Jalan Tembalang Selatan II dan kembali ke JNE, Jalan Kyai Saleh No. 10 dengan jarak minimum 61,9 Km dalam sekali tempuh.

5.2. Saran

Berdasarkan simpulan hasil penelitian, saran yang perlu disampaikan adalah sebagai berikut.

1. Permasalahan *Travelling Salesman Problem* (TSP) memungkinkan untuk dikembangkan menjadi sistem berbasis *web,* sehingga sistem lebih mudah untuk diakses oleh masyarakat umum atau perusahaan distribusi, agen travel, tukang pos, dan lain sebagainya sehingga dapat membantu dalam efisiensi biaya dan waktu.

2. Untuk penelitian selanjutnya, diharapkan aplikasi yang digunakan dapat langsung terkoneksi dengan *Google Maps* guna mempermudah dalam pengambilan data jarak antar alamat.

DAFTAR PUSTAKA

Basuki, A. 2003a. Strategi Menggunakan Algoritma Genetika. Tersedia di http://basuki.lecturer.pens.ac.id/lecture/StrategiAlgoritmaGenetika.pdf [diakses 21-5-2015].

Basuki. A. 2003b. Algoritma Genetika Suatu Alternatif Penyelesaian Permasalahan Searching, Optimasi dan Machine learning. Tersedia di http://basuki.lecturer. pens.ac.id/lecture/AlgoritmaGenetika.pdf [diakses 21-5-2015].

Budayasa, I. K. 2007. *Teori Graf dan Aplikasinya*. Surabaya: Unesa University Press.

Desiani, A., & M. Arhami. 2006. *Konsep Kecerdasan Buatan*. Yogyakarta: Andi Offset.

Firmansyah, A. 2007. *Dasar-dasar Pemograman MATLAB*. IlmuKomputer.com.

Hopgood, A.A. 2001. *Intelligent Systems for Engineers and Scientists*,Bosca Raton: CRC Press LLC.

Iqbal M. 2009. *Dasar Pengolahan Citra Menggunakan MATLAB*. Departmen Ilmu dan Teknologi Kelautan IPB.

Khan, H. F. 1999. Solving TSP Problem Using Genetic *Algorithm. International Journal of Basic and Applied Science IJBAS*, volume 9, number 10.

Kusumadewi. S. 2003. *Artifical Intelligence:Teknik dan Aplikasinya*. Yogyakarta: Graha Ilmu.

Munir, R. 2005. *Matematika Diskrit*. Bandung : CV Informatika.

Munir, R. 2010. *Matematika Diskrit (4^{th} ed.)*. Bandung : CV Informatika.

Philip A, A.A. Taofiki & O. Kehinde. 2011. A Genetic Algorithm for Solving Travelling Salesman Problem. *International Journal of Advanced Computer Science and Applications (IJACSA).* Tersedia di https://thesai.org/ Downloads/Volume2No1/Paper%204-A%20Genetic%20Algorithm%20for%20 Solving%20Travelling%20Salesman%20Problem.pdf [diakses 01-08-2015].

Purnamasari, R.W, Dwijanto, & E. Sugiharti. 2013. Implementasi Jaringan Syaraf Tiruan Backpropagation Sebagai Sistem Deteksi Penyakit Tuberculosis. *Unnes Journal of Mathematics.* Tersedia di http://journal.unnes.ac.id/sju/index.php/ujm/article/view/3247/2988 [diakses 14-05-2015].

Purwananto, Y., D. Purwitasari, & A. W. Wibowo. 2005. Implementasi dan Analisis Algoritma Pencarian rute terpendek di kota Surabaya. *Jurnal Penelitian dan Pengembangan TELEKOMUNIKASI*, Vol 10 No.2, 94-101. Tersedia di http://www.researchgate.net/profile/Diana_Purwitasari/publication/260302626_IMPLEMENTASI_DAN_ANALISIS_ALGORITMA_PENCARIAN_RUTE_TERPENDEK_DI_KOTA_SURABAYA/links/53ebcaae0cf250c8947c6458.pdf?inViewer=1&disableCoverPage=true&origin=publication_detail [diakses 21-1-2015].

Saptono, F.& T. Hidayat. 2007. Perancangan Algoritma Genetika Untuk Menentukan Jalur Terpendek. *Seminar Nasional Aplikasi Teknologi Informasi.* Yogyakarta: Universitas Islam Indonesia.

Sari, F.A., Dwijanto, & E. Sugiharti. 2013. Implementasi Algoritma Genetika Untuk Menyelesaikan Travelling Salesman Problem. *UNNES Journal of Matematics*, Vol. 2, No.2, Nopember 2013. Tersedia di http://journal.unnes.ac.id/sju/ index.php/ujm/article/view/3251 [22-1-2015].

Siang, J. J. 2002. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Yogyakarta: ANDI.

Suharto,I., B. Girisuta, & A. Miryanti. 2004. *Perekayasaan Metodologi Penelitian*. Yogyakarta: Andi.

Sutanto, J., S. R. Hendrawan & Y. Kurniawan. 2011. *Algoritma Branch and Bound untuk Masalah Penjadwalan Mesin Paralel*. Bandung: Laboratorium Ilmu dan Komputasi Departemen Teknik Informatika, Institut Teknologi Bandung. Tersedia di http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/Makalah/ MakalahStmik04.pdf [diakses 21-1-2015].

Wibowo, M.A.2009. *Aplikasi Algoritma Genetika Untuk Penjadwalan Mata Kuliah*. Semarang: Jurusan Matematika Fakultas Sains dan Matematika UNDIP.

Widyawati, K., Mashuri, & R. Arifudin. 2014. Analisis Algoritma Branch and Bound Untuk Menyelesaikan Masalah Penjadwalan Proyek Pembangunan Mega Tower. *UNNES Journal of Mathematics*, Vol. 1, No. 3, Mei 2014. Tersedia di http://journal.unnes.ac.id/sju/index.php/ujm/article/view/3283 [dikses 21-1-2015].

Zukhri, Z. 2014. Algoritma Genetika Metode Komputasi Evolusioner untuk Menyelesaikan Masalah Optimasi. Yogyakarta: Penerbit Andi.

Zulfikar, N. 2008. *Aplikasi Algoritma Genetika Untuk Mencari Rute Terpendek N-Buah Node*. Skripsi. FTIK Unikom.