PENERAPAN ALGORITMA *TABU SEARCH* UNTUK MENYELESAIKAN *VEHICLE ROUTING PROBLEM*

skripsi

disajikan sebagai salah satu syarat

untuk memperoleh gelar Sarjana Sains

Program Studi Matematika

oleh

Fajar Eska Pradhana

4150407007

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2011

PERNYATAAN

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, Agustus 2011

Fajar Eska Pradhana

4150407007

PENGESAHAN

Skripsi yang berjudul

Aplikasi Algoritma *Tabu Search* untuk Menyelesaikan *Vehicle Routing Problem*

disusun oleh

Fajar Eska Pradhana

4150407007

telah dipertahankan di hadapan sidang Panitia Ujian Skripsi FMIPA Unnes pada tanggal 25 Agustus 2011.

Panitia:

Ketua	Sekretaris
Dr. Kasmadi Imam S., M.S.	Drs. Edy Soedjoko, M.Pd
195111151979031001	195604191987031001

Ketua Penguji

Zaenal Abidin, S.Si., M.CS.

198205042005011001

Anggota Penguji/	Anggota Penguji/
Pembimbing Utama	Pembimbing Pendamping
Endang Sugiharti, S.Si, M.Kom	Muhammad Kharis, S.Si, M.Sc
197401071999032001	1982102200511001

PERSEMBAHAN

- Untuk ibu dan ayahku serta adik-adikku,
- Untuk Crew Math '07,
- Untuk teman-teman seperjuangan (Himatika 2009, KKN Ds Ngajaran Kec. Tuntang, Anida Kos), dan
- Untuk almamaterku.

MOTTO

- Sesungguhnya sesudah kesulitan ada kemudahan (QS Al-Insyirah : 6).
- Ilmu menunjukan kebenaran akal, maka barang siapa yang berakal, niscaya dia berilmu (Sayyidina Ali bin Abi Tholib).
- Sabar dalam mengatasi kesulitan dan bertindak bijaksana dalam mengatasinya adalah sesuatu yang utama (Fajar Eska Pradhana).

PRAKATA

Puji syukur penulis panjatkan kehadirat Allah SWT yang telah memberikan limpahan rahmat dan hidayah-Nya, sehingga skripsi yang berjudul ñPenerapan Algoritma Tabu Search Untuk Menyelesaikan *Vehicle Routing Problem*ò ini dapat terselesaikan. Sholawat dan salam senantiasa penulis tujukan kepada Nabi Muhammad SAW, yang kita nantikan syafaatnya kelak di yaumil akhir.

Skripsi ini tidak terlepas dari bimbingan, bantuan dan saran dari segala pihak dalam penulisannya. Oleh karena itu, dalam kesempatan ini penulis mengucapkan terima kasih kepada:

1. Prof. Dr. Sudijono Sastroatmodjo, M.Si., Rektor Universitas Negeri Semarang.
2. Dr. Kasmadi Imam S, M.S., Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Drs. Edy Soedjoko, M.Pd., Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
4. Endang Sugiharti, S.Si., M.Kom., Dosen Pembimbing Utama yang telah membimbing dan mengarahkan dalam penyusunan skripsi ini.
5. Muhammad Kharis, S.Si., M.Sc., Dosen Pembimbing Pendamping yang telah membimbing dan mengarahkan dalam penyusunan skripsi ini.
6. Seluruh dosen Jurusan Matematika FMIPA Universitas Negeri Semarang yang telah memberikan ilmu yang bermanfaat dan membantu kelancaran dalam penyusunan skripsi ini.

7. Kedua orang tua dan adik-adikku yang telah memberikan doa, dukungan, dan semangat yang tidak ternilai harganya sehingga penulis bisa menyelesaikan skripsi ini.
8. Semua pihak yang telah membantu penyusunan skripsi ini yang tidak dapat penulis sebutkan satu persatu.

Semoga Allah SWT memberi rahmat serta hidayah-Nya pada kita baik di dunia maupun di akhirat. Kesempurnaan hanya milik Allah Yang Maha Kuasa, penulis berharap skripsi ini dapat memberi manfaat bagi Almamater pada khususnya dan pembaca pada umumnya.

Semarang, Agustus 2011
Penulis

Fajar Eska Pradhana
4150407007

ABSTRAK

Pradhana, Fajar, Eska. 2011. *Penerapan Algoritma Tabu Search untuk Menyelesaikan Vehicle Routing Problem*. Skripsi, Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang. Pembimbing Utama Endang Sugiharti, S.Si., M.Kom. dan Pembimbing Pendamping Muhammad Kharis, S.Si., M.Sc.

Kata kunci: *Vehicle Routing Problem* (VRP), *The Classical Vehicle Routing Problem*, *tabu search*, heuristik.

VRP memiliki aplikasi yang penting di bidang manajemen distribusi. VRP merupakan permasalahan *integer programming* yang masuk kategori *NP-Hard Problem* (*Nondeterministik Polynomial – Hard*). *The Classical Vehicle Routing Problem* (CVRP) merupakan varian dasar pada VRP. Model masalah CVRP secara umum merupakan kunjungan tunggal dengan hanya satu kendaraan yang diperbolehkan mengunjungi pelanggan. Pada umumnya VRP terselesaikan dengan menggunakan berbagai variasi metode heuristik, salah satunya adalah algoritma *Tabu Search* (TS). Algoritma *Tabu Search* termasuk dalam teknik pencarian heuristik.

Penelitian dilakukan di IT COMM cabang Yogyakarta yang beralamat di Jl. Wonosari Km. 8 No. 99 Bantul. IT COMM mempunyai sejumlah subdistributor yang letaknya berpencar sehingga dapat digunakan sebagai studi kasus dalam tugas akhir ini.

Permasalahan yang diangkat pada penelitian ini adalah penentuan cara menyelesaikan VRP yaitu mencari jalur optimal untuk mendistribusikan barang pada perusahaan IT COMM menggunakan algoritma *Tabu Search* sehingga biaya transportasi minimum.

Berdasarkan pembahasan diperoleh simpulan bahwa penggunaan algoritma *tabu search* untuk menyelesaikan masalah VRP terdiri dari 6 langkah. Langkah pertama yaitu menentukan solusi awal pada iterasi 0 dan menetapkan nilai solusi awal sebagai nilai solusi optimum. Langkah kedua yaitu mencari solusi-solusi alternatif yang tidak melanggar kriteria tabu. Langkah ke tiga yaitu memilih solusi terbaik diantara solusi alternatif pada langkah ke dua. Langkah ke empat yaitu memilih nilai solusi optimum baru. Langkah ke lima yaitu memperbarui *tabu list* dengan memasukkan solusi optimum baru. Langkah ke enam yaitu apabila kriteria pemberhentian dipenuhi maka proses perhitungan berhenti dan diperoleh solusi optimum, jika tidak proses kembali berulang dimulai dari langkah ke dua. Dengan menggunakan data dari IT COMM, diperoleh hasil solusi optimum dengan rute Computa - ALNEC - IT COMM - WOW - WKM - Dian Kencana ï Saintech ï Fajar Aircond ï Surya I ï Rifani ï Larisa - Computa sepanjang 79 Km.

Berdasarkan pembahasan di atas, disarankan kepada Perusahaan IT COMM untuk menggunakan metode algoritma *Tabu Search* dalam proses distribusi sehingga biaya yang dikeluarkan minimal.

DAFTAR ISI

DAFTAR GAMBAR

Gambar Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR SIMBOL

Simbol	
$G(V, E)$	Graf G dengan himpunan titik V dan himpunan sisi E
$V(G)$	Himpunan titik pada graf G
$E(G)$	Himpunan sisi pada graf G
d_{ij}	Jarak antara titik i dan j
x_{ij}^{k}	Kendaraan k mengunjungi $j$setelah mengunjungi i
n	Banyak pelanggan (titik)
s, s^{*}, s^{0}	Solusi
S	Ruang solusi S
k	iterasi ke k
S^{k}	Himpunan solusi-solusi yang tidak tabu dalam $S(s^{k})$
$N(s), N(s^{k}) \subseteq S(s^{k})$	*Neighbourhood* solusi s
$f(s)$	Fungsi objektif

DAFTAR LAMPIRAN

Lampiran Halaman

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Proses distribusi yang optimal dalam sebuah industri, baik itu industri manufaktur maupun jasa, merupakan hal yang penting. Semakin mahal biaya distribusi berakibat naiknya harga produk sehingga memungkinkan terjadinya penurunan jumlah permintaan. Hal ini mengakibatkan pendapatan perusahaan tersebut menurun. Untuk meminimalisir keadaan tersebut, selain menekan biaya produksi perusahaan juga perlu menekan biaya distribusi.

Dalam proses distribusi, sebuah perusahaan yang mempunyai pabrik akan mengirimkan produknya ke berbagai distributor sebelum dapat digunakan oleh konsumen. Apabila proses distribusi bertujuan meminimalkan biaya, maka permasalahan tersebut dapat digolongkan dalam *Vehicle Routing Problem* (VRP).

VRP adalah sebuah masalah yang di dalamnya terdapat sejumlah rute untuk kendaraan yang berada pada satu atau lebih depot yang harus ditentukan jumlah kendaraanya agar tersebar merata secara geografis supaya bisa melayani konsumen. Setiap tujuan hanya boleh dilayani oleh satu kendaraan saja. Hal ini dilakukan dengan memperhatikan kapasitas kendaraan dalam satu kali angkut. VRP bertujuan untuk meminimalkan biaya yang biasanya berkaitan erat dengan jarak tempuh. Semakin jauh jarak tempuh maka konsumsi bahan bakar semakin banyak sehingga biaya yang dikeluarkan juga semakin besar untuk membeli

bahan bakar. Dalam penelitian ini kendaraan yang dipakai adalah kendaraan transportasi darat.

VRP merupakan permasalahan *integer programming* yang masuk kategori *NP-Hard Problem* (*Nondeterministik Polynomial – Hard*), yang berarti usaha komputasi yang digunakan akan semakin sulit dan banyak seiring dengan meningkatnya ruang lingkup masalah. Untuk masalah seperti ini biasanya yang dicari adalah aproksimasi solusi yang terdekat, karena solusi tersebut dapat dicari dengan cepat dan akurat.

VRP memiliki aplikasi yang penting di bidang manajemen distribusi, sehingga menjadi salah satu contoh masalah yang banyak dipelajari dalam literatur optimasi kombinatorial. Pada umumnya VRP terselesaikan dengan menggunakan berbagai variasi metode heuristik, salah satunya adalah algoritma *Tabu Search* (TS). Algoritma *Tabu Search* termasuk dalam teknik pencarian heuristik, yaitu suatu strategi untuk melakukan proses pencarian ruang keadaan (*state space*) suatu problema secara selektif.

Konsep dasar dari *Tabu Search* yaitu menuntun setiap tahapannya agar dapat menghasilkan kriteria aspirasi yang paling optimum tanpa terjebak ke dalam solusi awal yang ditemukan selama tahapan ini berlangsung. Maksud dari algoritma ini adalah mencegah terjadinya perulangan dan ditemukannya solusi yang sama pada suatu iterasi yang akan digunakan lagi pada iterasi selanjutnya.

IT COMM merupakan distributor yang dipilih oleh sistem jaringan panasonic Jepang untuk mendistribusikan produk teknologi komunikasi dan informasi di Indonesia. IT COMM pusat berada di Jakarta, sedangkan penelitian

dilakukan di kantor cabang Yogyakarta yang beralamat di Jl. Wonosari Km. 8 No. 99 Bantul. IT COMM cabang Yogyakarta mempunyai sejumlah subdistributor yang akan menyalurkan barang ke pelanggan. Letak dari subdistributor berpencar sehingga dapat digunakan sebagai studi kasus dalam tugas akhir ini.

Mengingat prinsip algoritma TS dalam menemukan jarak perjalanan paling pendek tersebut, maka TS merupakan salah satu metode yang dapat digunakan untuk menyelesaikan VRP. Algoritma ini disimulasikan pada perusahaan IT COMM *by* Panasonic Yogyakarta.

Sekarang ini, penerapan komputer untuk membantu mengerjakan tugas-tugas manusia sudah mencakup bidang yang sangat luas. Salah satu hal yang menjadikan perkembangan komputer begitu cepat adalah kemajuan di bidang pemrograman komputer. Salah satu perangkat lunak pemrograman komputer yang cukup banyak dipakai yaitu Borland Delphi.

Borland Delphi merupakan sarana pemrograman aplikasi visual. Dephi merupakan generasi penerus dari Turbo Pascal. Bahasa pemrograman yang dipakai adalah bahasa pemrograman Pascal atau yang kemudian disebut bahasa pemrograman Delphi. Kelebihan yang dimiliki Delphi antara lain *form* dan komponen-komponennya dapat dipakai ulang dan dikembangkan, menghasilkan file terkompilasi yang berjalan lebih cepat, serta kemampuan mengakses data dari bermacam-macam format (Wahana Komputer, 2003:2).

Berdasarkan latar belakang di atas penulis mengambil judul ñ*Penerapan Algoritma Tabu Search untuk Menyelesaikan Vehicle Routing Problem*ò dengan studi kasus pada perusahaan IT COMM *by* Panasonic cabang Yogyakarta.

1.2 Permasalahan

Berdasarkan latar belakang yang telah dijelaskan di atas, maka permasalahan yang dibahas dalam tugas akhir ini adalah penentuan cara menyelesaikan masalah VRP yaitu mencari jalur optimal untuk mendistribusikan barang pada perusahaan IT COMM menggunakan algoritma *Tabu Search* sehingga biaya transportasi minimum.

1.3 Batasan Masalah

Pembahasan dalam Tugas Akhir ini dibatasi hanya pada masalah VRP dengan asumsi-asumsi sebagai berikut.

(1) kendaraan yang dimaksud dalam penelitian ini adalah kendaraan transportasi darat berupa mobil *box*.

(2) banyaknya depot satu.

(3) jarak *dealer* tujuan dengan depot diketahui.

(4) jalur yang dilalui yaitu jalur dua arah.

(5) perhitungan dilakukan menggunakan program Borland Delphi 7.

1.4 Tujuan dan Manfaat

1.4.1 Tujuan

Tujuan dari pembahasan tugas akhir ini adalah.

(1) memahami Algoritma *Tabu Search*, dan

(2) menyelesaikan masalah VRP menggunakan Algoritma *Tabu Search*.

1.4.2 Manfaat

Manfaat dari pembuatan tugas akhir ini adalah.

(1) sebagai dasar dari pengembangan aplikasi-aplikasi lainnya yang menggunakan algoritma TS sebagai alat untuk memecahkan suatu *problem*,

(2) sebagai masukan untuk perusahaan dalam pengambilan keputusan tentang distribusi barang yang diproduksi.

1.5 Sistematika Penulisan

Secara garis besar, penulisan tugas akhir ini terdiri dari 3 (tiga) bagian, yaitu bagian awal, bagian isi, dan bagian akhir. Bagian awal tugas akhir berisi halaman judul, lembar pengesahan, persembahan, motto, prakata, abstrak, daftar isi, daftra gambar, daftar tabel, daftar simbol dan daftar lampiran.

Bagian isi terdiri dari 5 (lima) bab, yaitu sebagai berikut.

Bab I Pendahuluan meliputi latar belakang masalah, permasalahan, batasan masalah, tujuan dan manfaat, dan sistematika penulisan tugas akhir.

Bab II Landasan Teori meliputi konsep-konsep dasar Algoritma *Tabu Search* dan konsep dari *Vehicle Routing Problem*.

Bab III Metode Penelitian meliputi studi pustaka, metode pengumpulan data, perumusan masalah, pemecahan masalah, dan penarikan kesimpulan.

Bab IV Hasil Penelitian dan Pembahasan. Pada bab ini dikemukakan hasil penelitian dan pembahasan yang meliputi analisis penggunaan Algoritma

Tabu Search untuk menyelesaikan *Vehicle Routing Problem* pada perusahaan IT Comm *by* Panasonic Yogyakarta.

Bab V Penutup meliputi simpulan dan saran. Simpulan merupakan jawaban dari permasalahan yang ada. Saran berupa anjuran atau rekomendasi.

Pada bagian akhir, berisi daftar pustaka dan lampiran-lampiran yang mendukung tugas akhir.

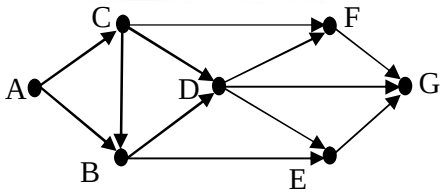
BAB 2
TINJAUAN PUSTAKA

2.1 Masalah Optimasi

Optimasi adalah proses pencarian satu atau lebih penyelesaian layak (*feasible*) yang berhubungan dengan nilai-nilai ekstrim dari satu atau lebih nilai objektif pada suatu masalah sampai tidak terdapat solusi ekstrim lain yang dapat ditemukan (Berlianty dan Miftahol, 2010:8). Optimasi memegang peranan penting dalam mendesain suatu sistem. Melalui optimasi, suatu sistem dapat mengeluarkan biaya yang lebih murah, mendapatkan keuntungan yang lebih tinggi, mempersingkat waktu proses dan optimalisasi yang lain.

2.2 Lintasan Terpendek

Masalah lintasan terpendek sebenarnya merupakan sebuah persoalan optimasi. Masalah lintasan terpendek dapat direpresentasikan menggunakan graf terhubung yang berbobot. Bobot dalam graf tersebut dapat berupa biaya, waktu, dan jarak. Oleh karena itu, masalah lintasan terpendek tidak hanya dilihat dari sudut pandang jarak saja. Namun inti dari masalah lintasan terpendek ini adalah menemukan bobot minimal dari lintasan yang dilalui dalam graf.



Gambar 2.1 Graf ABCDEFG

Berdasarkan gambar di atas, apabila dari *dealer* A ingin menuju *dealer* G maka dapat dipilih rute yang tersedia, yaitu.

A → C → F → G

A → C → D → F → G

A → C → D → G

A → C → D → E → G

A → C → B → D → F → G

A → C → B → D → G

A → C → B → D → E → G

A → C → B → E → G

A → B → D → F → G

A → B → D → G

A → B → D → E → G

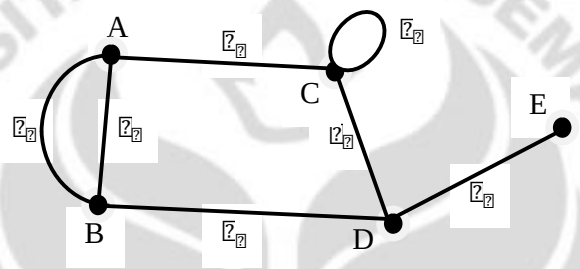
A → B → E → G

Berdasarkan data di atas, dapat dihitung jarak terpendek dari *dealer* A menuju *dealer* G dengan menghitung jarak antar *dealer* tersebut. Apabila jarak antar *dealer* belum diketahui, jarak dapat dihitung berdasarkan koordinat *dealer-dealer* tersebut dan dihitung jarak terpendek dari *dealer* A menuju *dealer* G.

2.3 Teori Graf

2.3.1 Definisi Graf

Sebuah graf $G = (V, E)$ adalah suatu sistem yang terdiri atas suatu himpunan objek $V = \{v_1, v_2, \ldots\}$ yang disebut himpunan titik, dan sebuah koleksi $E = \{e_1, e_2, \ldots\}$ yang merupakan koleksi sisi sedemikian hingga tiap sisi e_i dikaitkan dengan pasangan tak-terurut $\{v_1, v_2\}$. Titik v_1, v_2 yang berkaitan dengan e_i disebut titik-titik ujung sisi e_i (Sutarno,dkk., 2003:59).



Gambar 2.2 Graf G

Gambar 2.2 merupakan contoh graf dengan $V = \{A, B, C, D, E\}$ dan $E = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$.

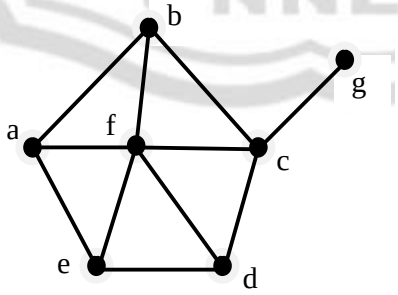
Dalam sebuah graf, dimungkinkan adanya sisi e_i dikaitkan dengan pasangan $\{v_1, v_2\}$. Seperti contoh di atas, sisi e_1 dikaitkan dengan pasangan $\{C, C\}$. Sisi yang dua titik ujungnya sama disebut *loop*/gelang. Sisi yang menghubungkan pasangan titik yang sama disebut sisi rangkap, contoh sisi e_2 dan e_3 menghubungkan pasangan titik yang sama yaitu $\{A, B\}$.

2.3.2 Definisi *Walk*, *Trail*, *Path* dan *Cycle*

Misalkan G adalah sebuah graf. Sebuah *walk* (jalan) di G adalah sebuah barisan berhingga (tak kosong) yang suku-sukunya bergantian titik dan sisi, sedemikian sehingga v_{i-1} dan v_i merupakan titik-titik akhir sisi e_i (Sutarno,dkk., 2003:65).

Misalkan jalan W adalah sebuah jalan dari v_0 ke v_k. Titik v_0 dan v_k berturut-turut disebut titik awal dan titik akhir. Sedangkan titik-titik $v_1, v_2, \ldots, v_{k-1}$ disebut titik-titik internal dari W dan k disebut panjang dari W. Perhatikan bahwa panjang jalan dari W adalah banyaknya sisi dalam W. Jika semua sisi $e_1, e_2, \ldots, e_k$ dalam jalan W berbeda, maka W disebut *trail* (jejak). Jika semua titik dalam *trail* semuanya berbeda maka *trail* tersebut merupakan *path* (lintasan). Dengan kata lain *path* merupakan *walk* yang semua sisi dan titiknya berbeda (Sutarno,dkk., 2003:65).

Sebuah jalan W disebut tertutup jika titik awal dan titik akhir dari W identik (sama). Jejak tertutup dinamakan sirkuit (*close trail*). Sirkuit yang titik awal dan titik internalnya berbeda disebut *cycle* (siklus) (Sutarno,dkk., 2003:65).



Jalan (*walk*) : a f b f c d c

Jejak (*trail*) : a f c d f b

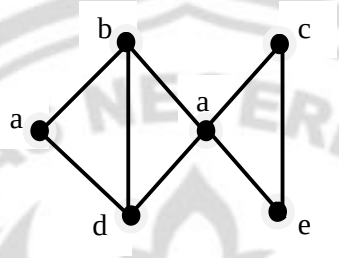
Lintasan (*path*) : a e f c g

Siklus (*cycle*) : a b c d e a

Gambar 2.3 Contoh *Walk*, *Trail*, *Path* dan *Cycle*

2.3.3 Definisi Graf Terhubung

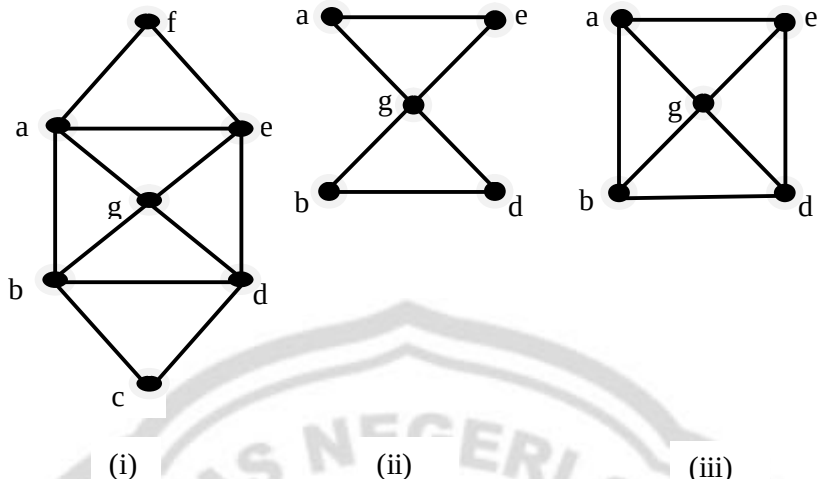
Sebuah graf dikatakan terhubung jika graf tersebut hanya terdiri atas satu bagian (satu komponen). Dalam graf terhubung G, untuk setiap pasang titik sembarang dan di G, terdapat suatu lintasan dari titik menuju titik (Sutarno,dkk., 2003:67).



Gambar 2.4 Contoh Graf Terhubung

2.3.4 Definisi Graf Euler dan Graf Hamilton

Sebuah sirkuit di graf G yang memuat semua sisi G disebut sirkuit Euler. Sebuah graf yang memuat sirkuit Euler disebut graf Euler. Graf yang memuat jejak euler disebut graf semi Euler Sebuah siklus yang memuat semua titik sebuah graf disebut siklus Hamilton. Graf yang memuat siklus Hamilton disebut graf Hamilton. Jadi graf terhubung G disebut graf Euler jika ada jejak (trail) tertutup yang memuat semua sisi pada graf G, sedangkan graf terhubung G disebut graf Hamilton jika ada sebuah siklus yang memuat semua titik pada garaf G itu. Graf yang memuat lintasan Hamilton disebut graf semi Hamilton (Sutarno,dkk., 2003:96).



Graf (i) merupakan graf Euler dan graf Hamilton.

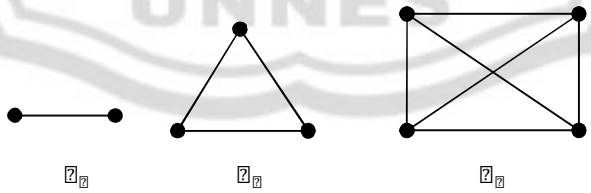
Graf (ii) merupakan graf Euler dan *trail* Eulernya aegbdga.

Graf (iii) merupakan graf Hamilton dengan *cycle* Hamiltonnya aegdba.

Gambar 2.5 Contoh Graf Euler dan Graf Hamilton

2.3.5 Definisi Graf Lengkap

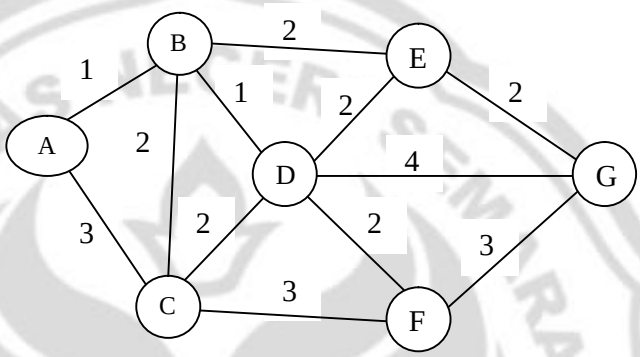
Misalkan $G = (V, E)$ adalah sebuah graf sederhana. Jika untuk setiap pasangan titik v_i dan v_j di G terdapat sebuah sisi yang menghubungkannya, maka G disebut graf lengkap. Graf lengkap dengan n titik dinotasikan dengan K_n (Sutarno,dkk, 2003:82).



Gambar 2.6 Contoh-contoh Graf Lengkap

2.3.6 Definisi Graf Berbobot

Sebuah graf G dikatakan graf berarah apabila setiap sisi dari graf G mempunyai arah yang ditunjukkan dengan tanda anak panah. Graf G dikatakan graf berbobot apabila semua sisinya mempunyai bobot/nilai (Sutarno,dkk, 2003:107).



Gambar 2.7 Contoh Graf Berarah Sekaligus Graf Berbobot

2.4 *Vehicle Routing Problem* (VRP)

VRP memiliki aplikasi yang penting di bidang manajemen distribusi, sehingga menjadi salah satu contoh masalah yang banyak dipelajari dalam literatur optimasi kombinatorial dan diakui sebagai salah satu pengalaman tersukses dalam riset operasi. Tipe VRP secara umum digambarkan sebagai suatu kasus di mana sejumlah kendaraan dengan kapasitas tertentu harus mengirim sejumlah barang dari suatu depot dengan asumsi jarak antara pelanggan telah diketahui sehingga tujuan dari masalah ini adalah meminimalkan jarak tempuh kendaraan supaya biaya operasional kendaraan minimal dengan berbagai pembatas.

Classical Vehicle Routing Problem (CVRP) merupakan varian dasar pada masalah rute kendaraan. Model masalah CVRP secara umum merupakan kunjungan tunggal dengan hanya satu kendaraan yang diperbolehkan mengunjungi pelanggan. Berbagai asumsi yang terdapat dalam CVRP antara lain: hanya terdapat satu depot, kapasitas kendaraan seragam, hanya terdapat satu komoditi yang didistribusikan, permintaan pelanggan telah diketahui sebelumnya dan kendala yang diperhitungkan hanya kapasitas kendaraan.

Pemodelan untuk CVRP memiliki parameter-parameter berikut: n adalah jumlah pelanggan, K menunjukkan kapasitas setiap kendaraan, q_i menunjukkan permintaan pelanggan i, c_{ij} adalah biaya perjalanan dari pelanggan i ke pelanggan j. Semua parameter dianggap nilai *integer* tidak negatif. Sekumpulan kendaraan homogen dengan kapasitas terbatas K dan sebuah depot utama, dengan indeks 0, melakukan perngiriman ke pelanggan, dengan indeks 1 sampai n.

Permasalahannya adalah menentukan rute pasti setiap kendaraan dimulai dan diakhiri di depot. Setiap pelanggan harus dipasangkan dengan tepat satu rute karena setiap pelanggan hanya bisa dilayani oleh satu kendaraan. Jumlah seluruh permintaan pelanggan yang ada pada setiap rute harus berada dalam batas kapasitas kendaraan. Tujuannya adalah untuk meminimalkan total biaya perjalanan.

Model matematika digambarkan pada sebuah graf $G(N, A)$. Sekumpulan titik/*node* N mewakili sekumpulan pelanggan C dari 1 sampai n dengan penambahan depot sebagai titik 0. Kumpulan busur A terdiri dari hubungan antar titik/*node* yang mungkin. Sebuah hubungan antara setiap dua titik/*node* pada graf

akan dimasukkan dalam A ini. Setiap busur (i, j), $i \neq j$, memiliki biaya perjalanan c_{ij} yang berhubungan dengannya.

Diasumsikan biaya simetris, contoh $c_{ij} = c_{ji}$. Kumpulan kendaraan seragam adalah V. Kendaraan-kendaraan tersebut memiliki kapasitas K dan semua pelanggan mempunyai permintaan q_i. Satusatunya variabel keputusan adalah x_{ij}

Formulasi untuk VRP sebagai berikut:

Minimal $\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{v} c_{ij} x_{ijk}$ (2.1)

kendala $\sum_{j=1}^{n} \sum_{k=1}^{v} x_{ijk} = 1$ $i = 2, \ldots, n$ (2.2)

$\sum_{i=1}^{n} q_i \sum_{j=1}^{n} x_{ijk} \leq K$ $k = 1, \ldots, v$ (2.3)

$\sum_{j=1}^{n} x_{1jk} = 1$ $k = 1, \ldots, v$ (2.4)

$\sum_{i=1}^{n} x_{ijk} - \sum_{i=1}^{n} x_{jik} = 0$ $j = 1, \ldots, n$; $k = 1, \ldots, v$ (2.5)

$x_{ijk} \in \{0,1\}$ $i, j = 1, \ldots, n$ dan $k = 1, \ldots, v$ (2.6)

(2.1) Menyatakan fungsi tujuan yang dioptimalkan.

(2.2) Menyatakan kendala penugasan yang menjamin tiap pelanggan dikunjungi tepat satu kali oleh tepat satu kendaraan.

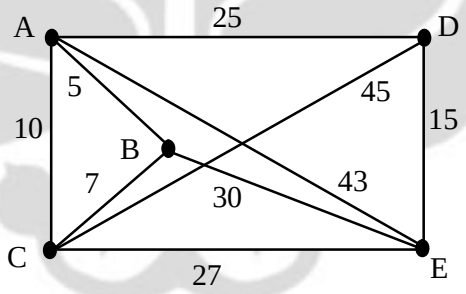
(2.3) Menyatakan kendala kapasitas yang menjamin muatan tidak melebihi kapasitas.

(2.4) dan (2.5) Menyatakan batasan jalur

(2.6) menyatakan suatu nilai biner (Satria,dkk, 2004:6-7).

Berikut adalah contoh kasus VRP: ñSuatu *dealer* A memiliki 4 buah tempat wisata. Sebuah biro *tour and travel* bermaksud untuk meluncurkan promosi untuk menarik wisatawan agar berkunjung ke tempat wisata tersebut. Biro tersebut menawarkan sebuah paket perjalanan wisata untuk mengunjungi semua tempat wisata tersebut. Untuk memprediksi biaya perjalanan, biro tersebut perlu mengetahui paling sedikit jumlah total jarak yang ditempuh selama perjalanan, dengan asumsi berangkat dari satu tempat dan kembali di tempat semulaò.

Ilustrasi dari permasalahan di atas disajikan dalam Gambar 2.8.



Gambar 2.8 Ilustrasi masalah VRP

Apabila contoh permasalahan di atas diubah ke dalam bentuk graf, maka permasalahan di atas adalah menentukan sirkuit Hamilton yang memiliki bobot minimum pada graf tersebut.

2.5 Algoritma *Tabu Search* (TS)

Tabu Search pertama kali diperkenalkan oleh Glover pada tahun 1986. *Tabu Search* merupakan salah satu algoritma yang berada dalam ruang lingkup metode heuristik. Konsep dasar dari *Tabu Search* adalah suatu algoritma yang menuntun setiap tahapannya agar dapat menghasilkan fungsi tujuan yang paling optimum tanpa terjebak ke dalam solusi awal yang ditemukan selama tahapan ini berlangsung. Tujuan dari algoritma ini adalah mencegah terjadinya perulangan dan ditemukannya solusi yang sama pada suatu iterasi yang akan digunakan lagi pada iterasi selanjutnya.

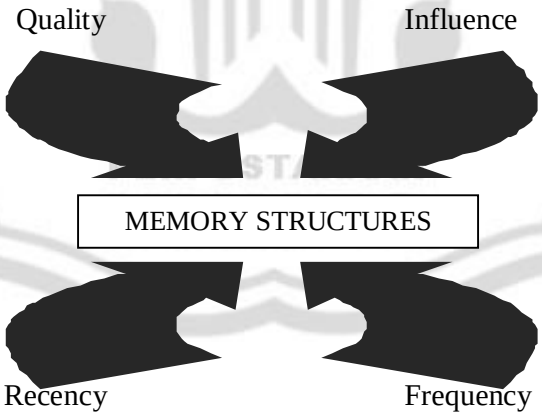
Menurut Glover dan Laguna (1997) kata tabu atau ñ*taboo*ò berasal dari bahasa Tongan, suatu bahasa Polinesia yang digunakan oleh suku Aborigin pulau Tonga untuk mengindikasikan suatu hal yang tidak boleh ñdisentuhò karena kesakralannya. Menurut kamus Webster, tabu berarti larangan yang dipaksakan oleh kebudayaan sosial sebagai suatu tindakan pencegahan atau sesuatu yang dilarang karena berbahaya. Bahaya yang harus dihindari dalam *Tabu Search* adalah rute perjalanan yang tidak layak, dan terjebak tanpa ada jalan keluar.

Untuk menunjang sistematis dari tujuan *Tabu Search* digunakan dua macam *tools*, yaitu *adaptive memory* and *responsive exploration*. Keutamaan dari *adaptive memory* menuntun suatu prosedur yang mampu melakukan pencarian solusi dengan lebih ekonomis dan efektif. *Responsive exploration* lebih menekankan pada tahapan tiap proses yang harus dilalui selama proses pencarian itu berlangsung, di mana pada setiap tahapan tersebut mempunyai suatu variabel

keputusan yang akan menuntun pada tahapan berikutnya sampai akhir proses pencarian dihentikan.

Struktur memori dalam *Tabu Search* menggunakan empat prinsip utama: *recency*, frekuensi, *quality*, dan *influence*. *Recency* atau lebih lengkapnya *recency based memory*, menjaga rekaman atau jejak solusi yang mengalami transformasi dan menyimpannya ke dalam suatu *short term memory* yang disebut *tabu list*. *Recency* menyediakan sebuah tipe informasi yang telah direkam oleh *recency based memory*. *Recency* dan frekuensi dapat saling melengkapi untuk membentuk suatu informasi permanen guna mengevaluasi pergerakan/*move* yang terjadi.

Quality menyatakan kemampuan untuk membedakan solusi terbaik yang dikunjungi selama pencarian atau iterasi berlangsung. *Influence* mempertimbangkan efek yang terjadi dari pemilihan solusi yang dipilih selama pencarian berlangsung, tidak hanya kualitas saja yang dipertimbangkan melainkan juga strukturnya.



Gambar 2.9 Struktur memori *Tabu Search*

Glover dan Laguna (1997) mengatakan bahwa memori pada *Tabu Search* mempunyai dua sifat yaitu *Explicit memory* dan *Attributive memory*. *Explicit memory* menyimpan *complete solution* yang umumnya menghabiskan alokasi ruang memori dan waktu, sehingga untuk menghindari hal ini *complete solution* dikurangi sehingga hanya terdiri dari *elite solution* yang dikunjungi selama pencarian. *Attributive memory* menyimpan informasi tentang atribut dari solusi yang ditemukan yang mungkin dapat berubah dari satu solusi ke solusi lain.

Sebagai suatu algoritma, *Tabu Search* mempunyai tahapan-tahapan dalam mencari solusi optimalnya. Tahapan-tahapan algoritma *Tabu Search* secara umum sebagai berikut:

Langkah 1 : Pilih solusi awal □ □ □. Tentukan □□ □ □ dan □ □ 0.

Langkah 2 : Tentukan □ □ □ □ 1 dan bentuk sebuah *subset* solusi □□ □ □□□, □□ yang tidak melanggar kriteria tabu atau paling sedikit salah satu kriteria terpenuhi.

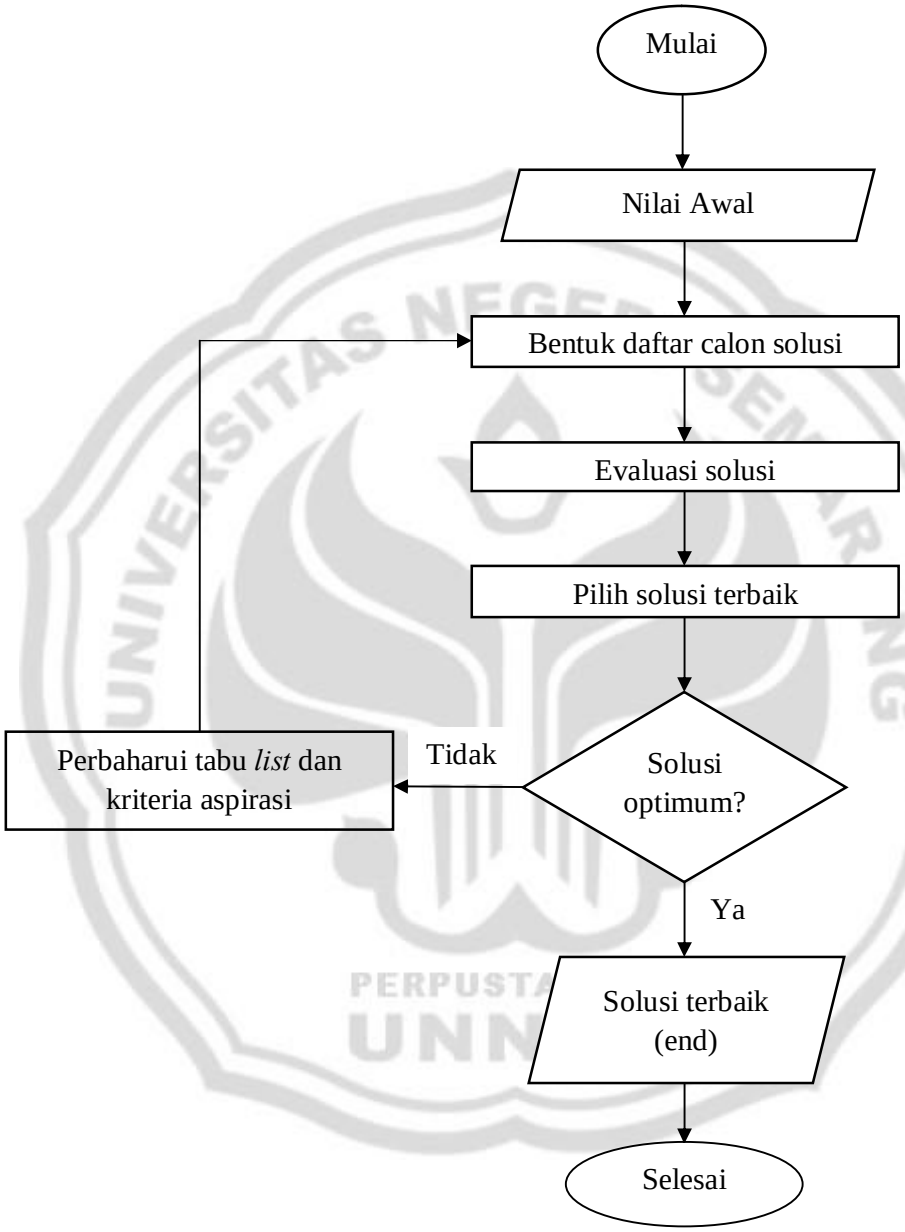
Langkah 3 : Pilih □□ □ □□ yang terbaik. Tentukan □ □ □□.

Langkah 4 : Jika □□□□□ □ □□□□□ maka tentukan □□ □ □□.

Langkah 5 : Perbaharui kriteria tabu dan kriteria aspirasi.

Langkah 6 : Jika ditemui *stopping condition* maka berhenti. Jika tidak, kembali ke Langkah 2 (Hertz, *et all*, 2002).

Berikut ini merupakan flowchart algoritma Tabu Search standar:



Gambar 2.10 *Flowchart* standar algoritma *Tabu Search*

2.5.1 Algoritma *Tabu Search* untuk menyelesaikan VRP

Algoritma *Tabu Search* untuk menyelesaikan masalah VRP dibuat dengan memperhatikan hal-hal sebagai berikut.

1. Representasi Solusi
2. Pembentukan inisial solusi (Solusi awal)
3. *Neighbourhood*
4. *Tabu List*
5. *Aspiration Criteria* (Kriteria Aspirasi)
6. *Termination Criteria* (Kriteria Pemberhentian)

2.5.1.1 Representasi Solusi

Representasi solusi yang digunakan dalam algoritma *tabu search* untuk VRP dalam tugas akhir ini adalah solusi fisibel yang ditulis sebagai suatu barisan titik-titik (*array*) dengan tiap titik tampak hanya satu kali dalam urutan. Titik yang dimaksud menunjukkan *dealer* (subdistributor). Untuk titik yang pertama dan terakhir adalah titik atau *dealer* 1 (depot), meskipun dalam representasi solusi titik 1 tidak harus selalu berada pada barisan pertama ataupun terakhir, karena rute yang terbentuk adalah suatu *cycle*.

2.5.1.2 Pembentukan Inisial Solusi (Solusi Awal)

Langkah awal yang dilakukan untuk menyelesaikan VRP menggunakan algoritma TS adalah membentuk solusi awal. Setiap solusi atau rute ditulis dalam bentuk *array* satu dimensi. Solusi awal dibentuk secara heuristik dengan mencari

titik yang terdekat dengan depot dan menambahkan titik pada rute sepanjang tidak membentuk *cycle*, begitu seterusnya hingga semua titik dikunjungi.

2.5.1.3 Neighbourhood

Dalam pencarian dengan teknik ini, setiap kemungkinan atribut dari struktur dapat dipindah-pindah. Perubahan yang dipakai oleh dua *neighbourhood* dengan melakukan *swap* elemen matriks atau kombinasi elemen itu dengan menukar elemen lain dalam matriks. Misalkan X adalah himpunan semua solusi, ⍰⍰ ⍰ X, N⍰⍰⍰ X adalah himpunan *neighbourhood* solusi ⍰. Proses pencarian solusi bergerak dari solusi satu ke solusi selanjutnya dengan cara memilih solusi dalam solusi *neighbourhood* yang sudah ada yang tidak tergolong solusi terlarang ⍰V⍰ ⍰ N⍰⍰⍰untuk setiap iterasi.

Solusi *neighbourhood* dalam tugas akhir ini didefinisikan sebagai solusi alternatif yang diperoleh dengan melakukan *move* atau *swap*. Solusi *neighbourhood* diperoleh dengan menukarkan dua titik yang berada dalam solusi. Hal ini menjamin bahwa solusi yang terbentuk adalah solusi fisibel (Gendreau and Potvin, 2010:45).

2.5.1.4 Tabu List

Untuk menghindari terulangnya langkah yang diambil, maka dilakukan *tabu test* dengan menggunakan *tabu list* yang sudah ada. *Tabu list* berisi atribut solusi-solusi yang telah dikunjungi sebelumnya. Tujuan utama dari *tabu list* bukan untuk mencegah terulangnya langkah yang telah diambil, namun lebih kepada

agar tidak berjalan mundur. Situasi perulangan jarang sekali terjadi karena telah dikombinasikan dengan beberapa *neighbourhood* sehingga kemungkinan perulangan solusi yang telah dikunjungi hampir tidak mungkin.

Tabu Search menggunakan *tabu list* untuk menolak solusi-solusi yang memenuhi atribut untuk mencegah terjadinya *cycling* dalam proses pencarian solusi pada daerah yang sama dan menuntun proses pencarian untuk menelusuri daerah solusi yang berikutnya.

Ukuran *tabu list* untuk menghasilkan kualitas solusi yang baik akan bertambah seiring dengan membesarnya ukuran masalah. Namun, tidak ada aturan baku untuk menentukan ukuran *tabu list*. Hal ini disebabkan ukuran *tabu list* bergantung pada ketatnya kriteria tabu yang diterapkan. Ukuran *tabu list* yang terlalu panjang akan mengakibatkan buruknya kualitas solusi karena terlalu banyak *move* yang dilarang. Ukuran *tabu list* dalam tugas akhir ini diatur sedemikian rupa sehingga panjangnya sama dengan banyaknya iterasi yang telah ditetapkan sebelumnya (Glover dan Kochenberger, 2003).

2.5.1.5 Aspiration Criteria

Walaupun mempunyai peran sentral dalam TS, status tabu terkadang sangat kuat antara lain, tabu dapat melarang *move* yang atraktif, bahkan ketika tidak terdapat bahaya *cycling* atau status tabu mungkin mengarah pada stagnasi dalam proses pencarian. Oleh karena itu, diperlukan suatu metode untuk membatalkan status tabu tersebut yang disebut *aspiration criteria* (Glover dan Kochenberger, 2003).

Aturan dasar yang digunakan dalam kriteria aspirasi pada algoritma TS dalam tugas akhir ini adalah kualitas solusi terbaik dalam *neigbourhood* dan solusi yang terbentuk tidak sama dengan solusi yang sudah ada. Jika kualitas solusi baru dalam *neigbourhood* lebih baik dibanding dengan yang dicapai sebelumnya, maka solusi baru tersebut dicatat pada list sebagai solusi terbaik yang baru dan status tabu dicabut. Apabila kualitas solusi baru tidak lebih baik dari solusi sebelumnya maka solusi tersebut tetap dimasukkan dalam list tetapi status tabu tetap berlaku pada solusi tersebut. Kemudian proses pencarian dilanjutkan sampai kriteria pemberhentian (*termination criteria*) dipenuhi.

2.5.1.6 Termination Criteria

Dalam teori, pencarian dapat dilakukan terus kecuali bila nilai optimal dari masalah yang diselesaikan sudah diketahui sebelumnya. Pada praktiknya pencarian dihentikan dengan sungguh-sungguh pada beberapa titik. Kriteria pemberhentian yang biasa digunakan dalam TS adalah.

(1) setelah semua iterasi yang telah ditetapkan sebelumnya terpenuhi;

(2) setelah beberapa iterasi tanpa ada perbaikan pada nilai fungsi objektif;

(3) ketika fungsi objektif mencapai nilai batas atas atau nilai batas bawah yang telah ditentukan sebelumnya;

(4) ketika tidak ada lagi solusi baru yang dapat dibangkitkan dari *current neighbourhood solution* dimana semua *move* terdapat dalam *tabu list*.

Kriteria pemberhentian (*termination criteria*) yang dipakai dalam tugas akhir ini yaitu setelah semua iterasi yang telah ditentukan terpenuhi. Jumlah

iterasi yang dipilih yaitu sama dengan banyaknya *dealer* karena jumlah iterasi maksimal sama dengan panjang *tabu list*.

2.5.1.7 Intensification dan Diversification

Pada umumnya, *intensification* didasarkan pada beberapa *intermediate-term memory* seperti *recency memory* yang menyimpan beberapa iterasi dari bermacam-macam komponen solusi. Intensifikasi sering digunakan dalam TS, tetapi tidak selalu dibutuhkan. Hal ini dikarenakan terdapat bermacam-macam situasi dimana pencarian dilakukan dengan proses pencarian normal yang cukup teliti. Sehingga tidak membutuhkan banyak waktu untuk menyelidiki bagian dari ruang pencarian.

Strategi intensifikasi mempunyai tujuan untuk mengidentifikasi himpunan dari solusi elit sebagai dasar untuk mendapatkan atribut yang bagus ke dalam solusi yang baru dibuat.

Diversification didasarkan pada beberapa bentuk dari *long-term memory* seperti *frequency memory* yang menyimpan semua iterasi (dimulai dari awal pencarian) dari bermacam-macam komponen solusi. Strategi diversifikasi lebih memfokuskan pada eksplorasi di dalam ruang solusi yang baru.

Terdapat dua teknik utama dalam diversifikasi. Teknik pertama yaitu *restart diversification*, mengulangi proses pencarian solusi dari solusi yang berbeda yang dibangun secara acak. Teknik yang kedua yaitu *continuous diversification*, menggunakan informasi yang diperoleh dari proses pencarian sebelumnya untuk menemukan *moves* dengan jarak yang lebar.

Dalam skema tabu yang kompleks, pencarian dihentikan setelah melengkapi rangkaian tahapan-tahapan, durasi dari tiap-tiap tahap ditentukan oleh salah satu kriteria di atas.

2.5.2 Langkah-langkah Algoritma TS dalam VRP

(1) Menentukan solusi awal dan menetapkannya sebagai solusi optimum.

(2) Menentukan solusi alternatif yaitu dengan melakukan *move* (menukarkan) dua titik dalam solusi.

(3) Mengevalusi solusi-solusi alternatif dengan *tabu list* untuk melihat apakah kandidat solusi (solusi alternatif) tersebut sudah ada pada *tabu list*. Apabila solusi alternatif sudah ada dalam *tabu list*, maka solusi alternatif tersebut tidak akan dievaluasi lagi. Apabila solusi alternatif belum terdapat dalam *tabu list*, maka solusi alternatif tersebut disimpan dalam *tabu list* sebagai solusi alternatif terbaik.

(4) Memilih solusi terbaik dan menetapkannya sebagai solusi optimum baru.

(5) Memperbarui *tabu list* dengan memasukkan solusi optimum baru.

(6) Apabila kriteria pemberhentian terpenuhi maka proses berhenti dan diperoleh solusi optimum. Jika tidak, proses kembali berulang dimulai dari langkah ke dua.

Contoh penggunaan algoritma TS

Contoh penyelesaian kasus transportasi mencari lintasan terpendek menggunakan algoritma TS :

Misalkan diketahui 6 kota dengan jarak antar kota disajikan dalam tabel di bawah ini (Poetra, 2010).

Tabel 2.1 Data Jarak

	Kota 1	Kota 2	Kota 3	Kota 4	Kota 5	Kota 6
Kota 1	0	20	21	25	30	36
Kota 2	20	0	25	21	36	30
Kota 3	21	25	0	10	11	18
Kota 4	25	21	10	0	18	11
Kota 5	30	36	11	18	0	20
Kota 6	36	30	18	11	20	0

Pada kasus ini jalur yang ditetapkan dimulai dari kota ke-5 dan berakhir di kota ke-2. Menggunakan *Tabu search* dengan maksimum iterasi sama dengan banyak kota yaitu 6, diperoleh:

Jalur awal :

5 1 3 4 6 2 Panjang jalur = 102

Iterasi ke-1 :

Tabu List :

5 1 3 4 6 2 Panjang jalur = 102

Tetangga (Jalur alternatif berikutnya) :

Jalur ke-1 : 5 3 1 4 6 2 Panjang jalur = 98

Jalur terbaik sejauh ini = 98

Jalur ke-2 : 5 4 3 1 6 2 Panjang jalur = 115

Jalur ke-3 : 5 6 3 4 1 2 Panjang jalur = 93

Jalur terbaik sejauh ini = 93

Jalur ke-4 : 5 1 4 3 6 2 Panjang jalur = 113

Jalur ke-5 : 5 1 6 4 3 2 Panjang jalur = 112

Jalur ke-6 : 5 1 3 6 4 2 Panjang jalur = 101

Jalur terbaik sejauh ini = 93, yaitu pada jalur ke-3 diterima sebagai Jalur terbaik.

Jalur terbaik = 93

Iterasi ke-2 :

Tabu List :

5 1 3 4 6 2 Panjang jalur = 102

5 6 3 4 1 2 Panjang jalur = 93

Tetangga (Jalur alternatif berikutnya) :

Jalur ke-1 : 5 3 6 4 1 2 Panjang jalur = 85

Jalur terbaik sejauh ini = 85

Jalur ke-2 : 5 4 3 6 1 2 Panjang jalur = 102

Jalur ke-3 : 5 1 3 4 6 2 Panjang jalur = 102

Jalur ke-4 : 5 6 4 3 1 2 Panjang jalur = 82

Jalur terbaik sejauh ini = 82

Jalur ke-5 : 5 6 1 4 3 2 Panjang jalur = 116

Jalur ke-6 : 5 6 3 1 4 2 Panjang jalur = 105

Jalur terbaik sejauh ini = 82, yaitu pada jalur ke-4 diterima sebagai Jalur terbaik.

Jalur terbaik = 82

Iterasi ke-3 :

Tabu List :

5 1 3 4 6 2 Panjang jalur = 108

5 6 3 4 1 2 Panjang jalur = 93

5 6 4 3 1 2 Panjang jalur = 82

Tetangga (Jalur alternatif berikutnya) :

Jalur ke-1 : 5 4 6 3 1 2 Panjang jalur = 88

Jalur ke-2 : 5 3 4 6 1 2 Panjang jalur = 88

Jalur ke-3 : 5 1 4 3 6 2 Panjang jalur = 113

Jalur ke-4 : 5 6 3 4 1 2 Panjang jalur = 93

Jalur ke-5 : 5 6 1 3 4 2 Panjang jalur = 108

Jalur ke-6 : 5 6 4 1 3 2 Panjang jalur = 101

Jalur terbaik = 82

Seterusnya hingga 6 iterasi, dan pada iterasi ke-2 akan diperoleh suatu jalur terpendek, yaitu :

Jalur ke-4 : 5 6 4 3 1 2 Panjang jalur = 82

2.6 Borland Delphi 7

Borland Delphi atau yang biasa disebut Delphi saja merupakan sarana pemrograman visual yang menggunakan bahasa pemrograman Pascal atau yang kemudian juga disebut bahasa pemrograman Delphi. Delphi merupakan generasi penerus dari Turbo Pascal. Turbo Pascal diluncurkan pada tahun 1983 yang dirancang untuk dijalankan pada sistem operasi DOS, sedangkan Delphi diluncurkan pertama kali tahun 1995 yang dirancang untuk beroperasi di bawah sistem operasi Windows.

Perkembangan Delphi tidak berhenti sampai di situ. Setelah Delphi versi 1, diluncurkan pula Delphi versi 2 yang berjalan pada windows 95 atau Delphi 32 bit), Delphi versi 3 yang berjalan pada windows 95 ke atas dengan tambahan fitur internet atau web, dan berkembang terus hingga versi terkini yaitu Delphi 8. Delphi yang dipakai dalam tugas akhir ini adalah Delphi versi 7.

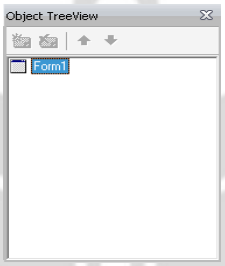
Delphi menyediakan fasilitas yang luas mulai dari fungsi untuk membuat form hingga fungsi untuk menggunakan beberapa format file basis data yang popular (dBASE, Paradox, dsb). Berikut merupakan beberapa kelebihan Delphi:

1. Komponen visual dan komponen non-visual sudah tersedia dalam Delphi. Komponen-komponen yang dapat ditemui antara lain komponen *Button*, komponen *Database*, komponen Menu dan Dialog.
2. Tersedia *template* aplikasi dan *template form*.
3. Memiliki lingkungan pengembangan visual yang dapat diatur sesuai kebutuhan.

4. Menghasilkan file terkompilasi yang berjalan lebih cepat.
5. Kemampuan mengakses data dari bermacam-macam format.

2.6.1 *Object Tree View*

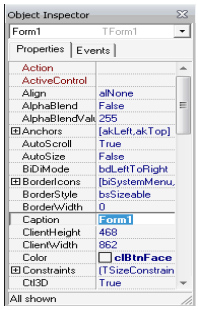
Merupakan sebuah diagram pohon yang menggambarkan hubungan logis menghubungkan semua komponen yang terdapat dalam suatu proyek program. Komponen tersebut meliputi *form*, modul atau *frame*. Fungsinya digunakan untuk menampilkan seluruh daftar komponen program dalam sebuah aplikasi program sesuai dengan penempatannya.



Gambar 2.11 *Object Tree View*

2.6.2 *Object Inspector*

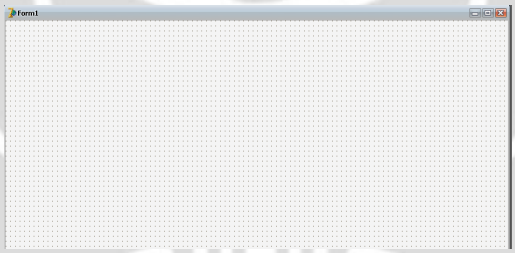
Merupakan jendela yang digunakan untuk mengatur tampilan komponen pada *form*, misal bagaimana mengubah tulisan *button* pada *command button* menjadi Simpan, atau menghapus tulisan pada label dan mengganti nama menjadi Nama Mahasiswa atau memberikan perintah tertentu pada sebuah komponen sehingga ada interaksi ketika program dijalankan.



Gambar 2.12 *Object Inspector*

2.6.3 *Form Designer*

Merupakan tempat yang digunakan untuk merancang semua aplikasi program yang diambil dari komponen *pallete.*



Gambar 2.13 *Form Designer*

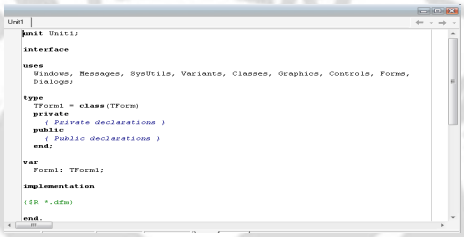
2.6.4 *Component Pallete*

Berisi sekumpulan objek yang akan digunakan dalam pembuatan program. Dalam komponen *pallete* semua *icon* dikelompokan dalam berbagai komponen sesuai dengan fungsi dan kegunaannya.

Gambar 2.14 *Component Pallete*

2.6.5 *Code Editor*

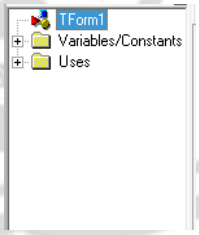
Bagian dari Delphi yang digunakan untuk menuliskan kode program. Pada bagian *code editor* terdapat 3 bagian utama yaitu bagian paling kiri yang berisi berupa angka menunjukan baris dan kolom. Keterangan *modified* menunjukan bahwa telah terjadi modifikasi terhadap baris program. Pada posisi paling kanan menunjukan status *keyboard* tentang tombol *insert* atau *over write*.



Gambar 2.15 *Code Editor*

2.6.6 *Code Explorer*

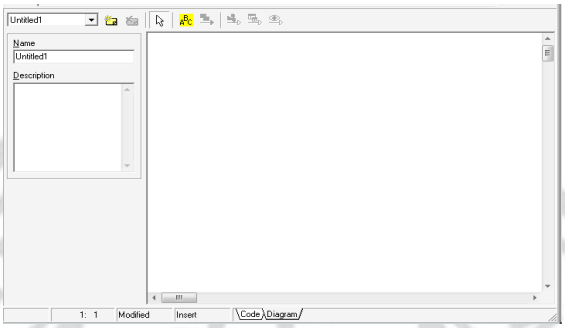
Jendela yang digunakan untuk menampilkan seluruh *variabel*, *type*, dan *routine* yang didefinisikan pada sebuah unit.



Gambar 2.16 *Code Explorer*

2.6.7 *Code Diagram*

Merupakan fasilitas pada Delphi yang digunakan untuk mendesain sebuah diagram atas komponen-komponen yang digunakan dalam satu rancangan aplikasi.



Gambar 2.17 *Code Diagram*

BAB 3
METODE PENELITIAN

3.1 Studi Pustaka

Studi pustaka merupakan penelaahan sumber pustaka yang berisi data-data dan informasi-informasi yang diperlukan dalam penelitian. Studi pustaka diawali dengan mengumpulkan sumber pustaka yang berupa buku-buku referensi, skripsi, makalah, *ebook* dan sebagainya yang berhubungan dengan kajian materi yang akan diteliti. Setelah sumber pustaka terkumpul dilanjutkan dengan penelaahan isi sumber pustaka tersebut sehingga muncul ide atau gagasan. Pada akhirnya sumber pustaka ini dijadikan landasan untuk melakukan penelitian.

3.2 Metode pengumpulan data

Penulis memperoleh data dengan metode dokumentasi yaitu metode pengumpulan data dengan cara mengambil data sekunder yang diperoleh dari IT COMM *by* Panasonic cabang Yogyakarta yang beralamat di Jl. Wonosari Km. 8 No. 99 Bantul yaitu berupa data banyaknya distributor dan jarak masing-masing distributor.

3.3 Perumusan Masalah

Dari hasil studi pustaka dan penelitian pada IT COMM *by* Panasonic cabang Yogyakarta, permasalahan yang muncul adalah penentuan cara menyelesaikan masalah VRP yaitu mencari jalur optimal untuk mendistribusikan barang pada

perusahaan IT COMM menggunakan algoritma *Tabu search* sehingga biaya transportasi minimum.

3.4 Pemecahan Masalah

Pada tahap ini dilakukan kajian pustaka, yaitu mengkaji permasalahan secara teoritis berdasarkan sumber-sumber pustaka yang relevan. Adapun langkah-langkah yang dilakukan dalam tahap pemecahan masalah ini adalah:

(1) Mempelajari teori dan materi tentang optimasi, graf, VRP dan algoritma *Tabu search*.

(2) Menerapkan langkah-langkah algoritma *Tabu search* dalam menyelesaikan masalah yang telah didapatkan.

(3) Membuat program algoritma *Tabu search* menggunakan *software* Borland Delphi 7.

3.5 Penarikan Kesimpulan

Langkah ini merupakan langkah terakhir dari penelitan. Penarikan kesimpulan didasarkan pada studi pustaka dan pembahasan permasalahan. Simpulan yang diperoleh merupakan hasil penelitian.

BAB 4
HASIL DAN PEMBAHASAN

4.1 Hasil Penelitian

IT COMM merupakan distributor yang dipilih oleh sistem jaringan Panasonic Jepang untuk mendistribusikan produk teknologi komunikasi dan informasi di Indonesia. IT COMM pusat berada di Jakarta, sedangkan penelitian dilakukan di kantor cabang Yogyakarta yang beralamat di Jl. Wonosari Km. 8 No. 99 Bantul. IT COMM cabang Yogyakarta mempunyai sejumlah subdistributor yang akan menyalurkan barang ke pelanggan. Berikut ini merupakan data subdistributor dari IT COMM:

Tabel 4.1 Data Subdistributor

No	Subdistributor	Alamat
1	Rifani	Jl. Parang 139
2	Dian Kencana	Jl. Lawu No 2
3	Fajar Aircond	Jl. Dr. Sutomo
4	Saintech	Jl. Prof. Dr. Yohannes
5	Surya Informatika	Jl. Prawirotaman
6	ALNEC	Jl. Janti 1
7	WOW	Jl. Raya Janti
8	WKM	Jl. Kaliurang Km 5
9	Larisa	Jl. Suryowijayan
10	Computa	Jl. Cik di tiro

Berdasarkan data yang diperoleh di atas, perusahaan IT COMM tidak mempergunakan metode khusus dalam pendistribusian. Produk pesanan diantar secara langsung secara bergiliran. Pengiriman dimulai dari depot yaitu IT COMM Yogyakarta menuju ke daerah tujuan. Proses distribusi dimulai dari subdistributor yang mempunyai jarak paling dekat dengan depot dan dilanjutkan dengan subdistributor lain yang jaraknya dekat dengan subdistributor terakhir. Pengiriman dilakukan sebanyak 3-5 kali dalam sebulan.

Tabel 4.2 Jarak antar subdistributor dan depot dalam Km.

***Dealer* Ke i**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
1	**0**	10	10	8	9	10	5	5	12	10	12
2	10	**0**	10	12	13	10	15	13	18	12	15
3	10	10	**0**	4	4	10	10	8	9	16	5
4	8	12	4	**0**	2	9	8	9	9	17	5
5	9	13	4	2	**0**	10	8	8	10	14	10
6	10	10	10	9	10	**0**	15	13	17	12	15
7	5	15	10	8	8	15	**0**	10	16	10	6
8	5	13	8	9	8	13	10	**0**	9	14	6
9	12	18	9	9	10	17	16	9	**0**	17	9
10	10	12	16	17	14	12	10	14	17	**0**	8
11	12	15	5	5	10	15	6	6	9	8	**0**

Keterangan:

1 = IT COMM	7 = ALNEC
2 = Rifani	8 = WOW
3 = Dian Kencana	9 = WKM
4 = Fajar Aircond	10 = Larisa
5 = Saintech	11 = Computa
6 = Surya Informatika	

Perusahaan IT COMM tidak menggunakan metode khusus dalam menentukan rute. Berdasarkan wawancara yang dilakukan, rute yang biasa dilalui untuk proses distribusi adalah 1 7 11 3 4 5 8 9 6 2 10 1 dengan jarak tempuh:

Jarak = 7 11 3 4 5 8 9 6 2 10 5 6 5 4 2 8 9 17 10 12 10 88 Km.

Pada bagian selanjutnya akan diperlihatkan hasil perhitungan manual dan pembahasan aplikasi dari penggunaan algoritma *Tabu search* diterapkan pada persoalan rute jalur perjalanan kendaraan pada IT COMM Yogyakarta yang akan diaplikasikan pada perangkat lunak menggunakan program Borland Delphi 7 sehingga akan diketahui keakuratan hasil dan lamanya waktu eksekusi dari algoritma tersebut.

4.1.1 Proses perhitungan manual berdasarkan algoritma *Tabu search*

Langkah 1

Langkah pertama yang dilakukan adalah memilih solusi awal dan menentukan solusi awal tersebut sebagai solusi optimum pada iterasi ke-0 (□ □ 0). Solusi awal ditentukan dengan mencari titik yang terdekat dengan depot dan menambahkan titik terdekat dengan titik sebelumnya pada rute sepanjang tidak membentuk *cycle*, begitu seterusnya hingga semua titik dikunjungi. Dengan metode tersebut diperoleh solusi awal yaitu jalur 1 □ 7 □ 11 □ 4 □ 5 □ 8 □ 3 □ 9 □ 6 □ 2 □ 10 dan secara otomatis solusi tersebut masuk dalam *tabu list* pada iterasi ke 0 sekaligus sebagai solusi optimum awal.

Langkah 2

Langkah ke-2 yaitu menentukan iterasi selanjutnya dan mencari solusi alternatif yang tidak melanggar kriteria tabu. Solusi alternatif diperoleh dengan menukar posisi dua titik atau *dealer* berdasarkan indeks. Banyak indeks sama dengan □□□□= 55.

Pada iterasi ke-0 diperoleh *tabu list* 1 □ 7 □ 11 □ 4 □ 5 □ 8 □ 3 □ 9 □ 6 □ 2 □ 10, maka solusi alternatif yang di dapat yaitu:

Jika indeks (1) maka posisi titik ke-1 ditukar dengan posisi titik ke-2, diperoleh jalur alternatif ke-1: 7 □ 1 □ 11 □ 4 □ 5 □ 8 □ 3 □ 9 □ 6 □ 2 □ 10.

Jika indeks (2) maka posisi titik ke-1 ditukar dengan posisi titik ke-3, diperoleh jalur alternatif ke-1: 11 □ 7 □ 1 □ 4 □ 5 □ 8 □ 3 □ 9 □ 6 □ 2 □ 10.

Jika indeks (3) maka posisi titik ke-1 ditukar dengan posisi titik ke-4, diperoleh jalur alternatif ke-1: 4 7 11 1 5 8 3 9 6 2 10.

é

Jika indeks (11) maka posisi titik ke-2 ditukar dengan posisi titik ke-3, diperoleh jalur alternatif ke-11: 1 11 7 4 5 8 3 9 6 2 10.

Jika indeks (12) maka posisi titik ke-2 ditukar dengan posisi titik ke-4, diperoleh jalur alternatif ke-12: 1 4 11 7 5 8 3 9 6 2 10.

Jika indeks (13) maka posisi titik ke-3 ditukar denagn posisi titik ke-5, diperoleh jalur alternatif ke-13: 1 5 11 4 7 8 3 9 6 2 10.

Begitu seterusnya hingga indeks mencapai indeks ke-55.

Langkah 3

Langkah selanjutnya yaitu memilih solusi terbaik di antara solusi alternatif yang telah didapat pada langkah 2. Solusi terbaik pada iterasi pertama diperoleh pada indeks (46), maka solusi tersebut dipilih sebagai solusi optimum sementara.

Langkah 4

Apabila nilai solusi terbaik pada Langkah ke-2 lebih kecil dari nilai solusi optimum awal, maka solusi optimum terbaik yang didapat dipilih sebagai solusi optimum. Pada Langkah 2 diperoleh solusi terbaik pada indeks (46) dengan nilai solusi 86. Karena nilai solusi terbaik lebih kecil dari nilai solusi optimum awal maka solusi terbaik pada Langkah 2 dipilih sebagai solusi optimum yang baru.

Langkah 5

Memperbarui *tabu list* dengan menambahkan rute solusi optimum yang diperoleh pada Langkah 4. Diperoleh *tabu list* baru yaitu:

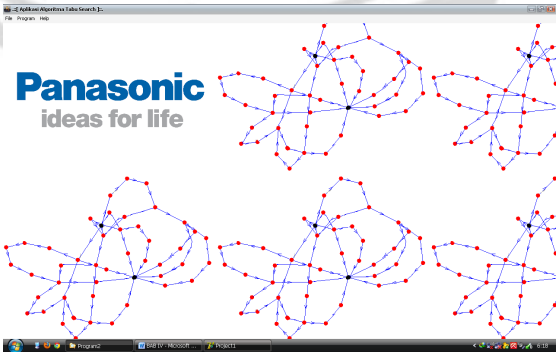
1. 1 7 11 4 5 8 3 9 6 2 10
2. 1 7 11 4 5 8 9 3 6 2 10

Langkah 6

Apabila kriteria pemberhentian dipenuhi maka proses berhenti. Jika tidak, proses diulang kembali mulai Langkah 2 dan akan berhenti ketika kriteria pemberhentian dipenuhi. Dalam tugas akhir ini kriteria pemberhentian yang dipakai yaitu setelah semua iterasi terpenuhi. Jumlah iterasi sama dengan banyaknya titik.

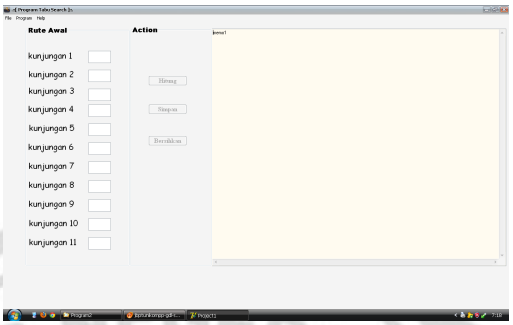
4.1.2 Proses perhitungan menggunakan program Borland Delphi 7

Untuk mempermudah perhitungan, maka model algoritma TS untuk masalah VRP yang telah diuraikan di atas diaplikasikan pada perangkat lunak menggunakan program Borlan Delphi 7 dan dieksekusi pada perangkat keras Intel (R) Core (TM)2 Duo CPU dengan processor 2,09 Ghz dan RAM 1,96 Gb. Hasil yang diperoleh adalah sebagai berikut.



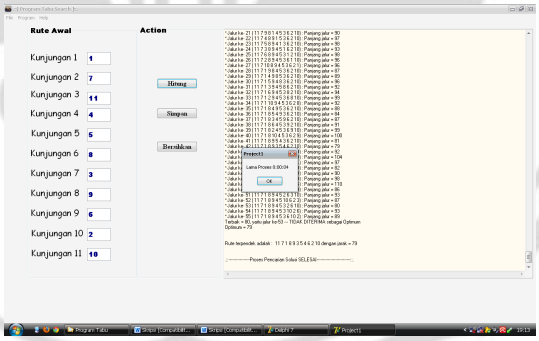
Gambar 4.1 *Form* Utama

Pada bagian *form* utama pilih menu Program *Tabu search*, maka akan muncul *form* Program *Tabu search* seperti pada Gambar 4.2.



Gambar 4.2 *Form* Program *Tabu search*

Untuk melakukan perhitungan, masukkan rute awal yang telah ditentukan sebelumnya dalam *form* Program *Tabu search*. Setelah semua rute dimasukkan, tekan *button* hitung maka proses akan berjalan. Setelah kriteria pemberhentian tercapai maka proses akan berhenti sehingga solusi optimal dan lama proses akan diketahui seperti pada Gambar 4.3.



Gambar 4.3 *Form Output*.

Output yang dihasilkan program di atas adalah sebagai berikut.

Algoritma T A B U S E A R C H

Iterasi ke- 1 :

Tabu list 1 : 1 7 11 4 5 8 3 9 6 2 10 Panjang jalur = 92

Solusi tersebut kemudian akan memasuki proses pencarian menggunakan algoritma TS dengan Solusi Optimum (Optimum) awal adalah 92. Kemudian jalur awal tersebut akan diubah untuk memperoleh solusi (jalur) alternatif lainnya.

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke- 1 (7 1 11 4 5 8 3 9 6 2 10) : Panjang jalur = 98

* Jalur ke- 2 (11 7 1 4 5 8 3 9 6 2 10) : Panjang jalur = 93

* Jalur ke- 3 (4 7 11 1 5 8 3 9 6 2 10) : Panjang jalur = 116

* Jalur ke- 4 (5 7 11 4 1 8 3 9 6 2 10) : Panjang jalur = 102

* Jalur ke- 5 (8 7 11 4 5 1 3 9 6 2 10) : Panjang jalur = 104

* Jalur ke- 6 (3 7 11 4 5 8 1 9 6 2 10) : Panjang jalur = 103

* Jalur ke- 7 (9 7 11 4 5 8 3 1 6 2 10) : Panjang jalur = 104

* Jalur ke- 8 (6 7 11 4 5 8 3 9 1 2 10) : Panjang jalur = 99

* Jalur ke- 9 (2 7 11 4 5 8 3 9 6 1 10) : Panjang jalur = 102

* Jalur ke- 10 (10 7 11 4 5 8 3 9 6 2 1) : Panjang jalur = 95

* Jalur ke- 11 (1 11 7 4 5 8 3 9 6 2 10) : Panjang jalur = 102

* Jalur ke- 12 (1 4 11 7 5 8 3 9 6 2 10) : Panjang jalur = 101

* Jalur ke- 13 (1 5 11 4 7 8 3 9 6 2 10) : Panjang jalur = 108

* Jalur ke- 14 (1 8 11 4 5 7 3 9 6 2 10) : Panjang jalur = 94

* Jalur ke- 15 (1 3 11 4 5 8 7 9 6 2 10) : Panjang jalur = 105

* Jalur ke- 16 (1 9 11 4 5 8 3 7 6 2 10) : Panjang jalur = 101

* Jalur ke- 17 (1 6 11 4 5 8 3 9 7 2 10) : Panjang jalur = 110

* Jalur ke- 18 (1 2 11 4 5 8 3 9 6 7 10) : Panjang jalur = 109

* Jalur ke- 19 (1 10 11 4 5 8 3 9 6 2 7) : Panjang jalur = 97

* Jalur ke- 20 (1 7 4 11 5 8 3 9 6 2 10) : Panjang jalur = 102

* Jalur ke- 21 (1 7 5 4 11 8 3 9 6 2 10) : Panjang jalur = 92

* Jalur ke- 22 (1 7 8 4 5 11 3 9 6 2 10) : Panjang jalur = 99

* Jalur ke- 23 (1 7 3 4 5 8 11 9 6 2 10) : Panjang jalur = 93

* Jalur ke- 24 (1 7 9 4 5 8 3 11 6 2 10) : Panjang jalur = 100

* Jalur ke- 25 (1 7 6 4 5 8 3 9 11 2 10) : Panjang jalur = 102

* Jalur ke- 26 (1 7 2 4 5 8 3 9 6 11 10) : Panjang jalur = 109

* Jalur ke- 27 (1 7 10 4 5 8 3 9 6 2 11) : Panjang jalur = 113

* Jalur ke- 28 (1 7 11 5 4 8 3 9 6 2 10) : Panjang jalur = 98

* Jalur ke- 29 (1 7 11 8 5 4 3 9 6 2 10) : Panjang jalur = 89

* Jalur ke- 30 (1 7 11 3 5 8 4 9 6 2 10) : Panjang jalur = 95

* Jalur ke- 31 (1 7 11 9 5 8 3 4 6 2 10) : Panjang jalur = 91

* Jalur ke- 32 (1 7 11 6 5 8 3 9 4 2 10) : Panjang jalur = 104

* Jalur ke- 33 (1 7 11 2 5 8 3 9 6 4 10) : Panjang jalur = 117

* Jalur ke- 34 (1 7 11 10 5 8 3 9 6 2 4) : Panjang jalur = 105

* Jalur ke- 35 (1 7 11 4 8 5 3 9 6 2 10) : Panjang jalur = 95

* Jalur ke- 36 (1 7 11 4 3 8 5 9 6 2 10) : Panjang jalur = 95

* Jalur ke- 37 (1 7 11 4 9 8 3 5 6 2 10) : Panjang jalur = 88

* Jalur ke- 38 (1 7 11 4 6 8 3 9 5 2 10) : Panjang jalur = 100

* Jalur ke- 39 (1 7 11 4 2 8 3 9 6 5 10) : Panjang jalur = 109

* Jalur ke- 40 (1 7 11 4 10 8 3 9 6 2 5) : Panjang jalur = 113

* Jalur ke- 41 (1 7 11 4 5 3 8 9 6 2 10) : Panjang jalur = 88

* Jalur ke- 42 (1 7 11 4 5 9 3 8 6 2 10) : Panjang jalur = 90

* Jalur ke- 43 (1 7 11 4 5 6 3 9 8 2 10) : Panjang jalur = 91

* Jalur ke- 44 (1 7 11 4 5 2 3 9 6 8 10) : Panjang jalur = 104

* Jalur ke- 45 (1 7 11 4 5 10 3 9 6 2 8) : Panjang jalur = 102

* Jalur ke- 46 (1 7 11 4 5 8 9 3 6 2 10) : Panjang jalur = 86

* Jalur ke- 47 (1 7 11 4 5 8 6 9 3 2 10) : Panjang jalur = 97

* Jalur ke- 48 (1 7 11 4 5 8 2 9 6 3 10) : Panjang jalur = 110

* Jalur ke- 49 (1 7 11 4 5 8 10 9 6 2 3) : Panjang jalur = 104

* Jalur ke- 50 (1 7 11 4 5 8 3 6 9 2 10) : Panjang jalur = 101

* Jalur ke- 51 (1 7 11 4 5 8 3 2 6 9 10) : Panjang jalur = 98

* Jalur ke- 52 (1 7 11 4 5 8 3 10 6 2 9) : Panjang jalur = 102

* Jalur ke- 53 (1 7 11 4 5 8 3 9 2 6 10) : Panjang jalur = 93

* Jalur ke- 54 (1 7 11 4 5 8 3 9 10 2 6) : Panjang jalur = 92

* Jalur ke- 55 (1 7 11 4 5 8 3 9 6 10 2) : Panjang jalur = 94

Terbaik = 86, yaitu jalur ke-46 --- DITERIMA sebagai Optimum

Optimum = 86

Jalur terpendek adalah jalur ke-46 dengan panjang 86, maka jalur tersebut terpilih untuk proses diversifikasi selanjutnya. Jalur tersebut ditambahkan ke

dalam *tabu list*. Karena panjang jalur tersebut lebih kecil dari Optimum sebelumnya, maka jalur tersebut dipilih sebagai Optimum yang baru.

Iterasi ke- 2 :

Tabu List :

1 1 7 11 4 5 8 3 9 6 2 10

2 1 7 11 4 5 8 9 3 6 2 10

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke- 1 (7 1 11 4 5 8 9 3 6 2 10) : Panjang jalur = 92

* Jalur ke- 2 (11 7 1 4 5 8 9 3 6 2 10) : Panjang jalur = 87

* Jalur ke- 3 (4 7 11 1 5 8 9 3 6 2 10) : Panjang jalur = 110

* Jalur ke- 4 (5 7 11 4 1 8 9 3 6 2 10) : Panjang jalur = 96

* Jalur ke- 5 (8 7 11 4 5 1 9 3 6 2 10) : Panjang jalur = 99

* Jalur ke- 6 (9 7 11 4 5 8 1 3 6 2 10) : Panjang jalur = 101

* Jalur ke- 7 (3 7 11 4 5 8 9 1 6 2 10) : Panjang jalur = 100

* Jalur ke- 8 (6 7 11 4 5 8 9 3 1 2 10) : Panjang jalur = 98

* Jalur ke- 9 (2 7 11 4 5 8 9 3 6 1 10) : Panjang jalur = 96

* Jalur ke- 10 (10 7 11 4 5 8 9 3 6 2 1) : Panjang jalur = 89

* Jalur ke- 11 (1 11 7 4 5 8 9 3 6 2 10) : Panjang jalur = 96

* Jalur ke- 12 (1 4 11 7 5 8 9 3 6 2 10) : Panjang jalur = 95

* Jalur ke- 13 (1 5 11 4 7 8 9 3 6 2 10) : Panjang jalur = 102

* Jalur ke- 14 (1 8 11 4 5 7 9 3 6 2 10) : Panjang jalur = 93

* Jalur ke- 15 (1 9 11 4 5 8 7 3 6 2 10) : Panjang jalur = 98

* Jalur ke- 16 (1 3 11 4 5 8 9 7 6 2 10) : Panjang jalur = 102

* Jalur ke- 17 (1 6 11 4 5 8 9 3 7 2 10) : Panjang jalur = 105

* Jalur ke- 18 (1 2 11 4 5 8 9 3 6 7 10) : Panjang jalur = 103

* Jalur ke- 19 (1 10 11 4 5 8 9 3 6 2 7) : Panjang jalur = 91

* Jalur ke- 20 (1 7 4 11 5 8 9 3 6 2 10) : Panjang jalur = 96

* Jalur ke- 21 (1 7 5 4 11 8 9 3 6 2 10) : Panjang jalur = 86

* Jalur ke- 22 (1 7 8 4 5 11 9 3 6 2 10) : Panjang jalur = 96

* Jalur ke- 23 (1 7 9 4 5 8 11 3 6 2 10) : Panjang jalur = 93

* Jalur ke- 24 (1 7 3 4 5 8 9 11 6 2 10) : Panjang jalur = 94

* Jalur ke- 25 (1 7 6 4 5 8 9 3 11 2 10) : Panjang jalur = 99

* Jalur ke- 26 (1 7 2 4 5 8 9 3 6 11 10) : Panjang jalur = 103

* Jalur ke- 27 (1 7 10 4 5 8 9 3 6 2 11) : Panjang jalur = 107

* Jalur ke- 28 (1 7 11 5 4 8 9 3 6 2 10) : Panjang jalur = 92

* Jalur ke- 29 (1 7 11 8 5 4 9 3 6 2 10) : Panjang jalur = 87

* Jalur ke- 30 (1 7 11 9 5 8 4 3 6 2 10) : Panjang jalur = 93

* Jalur ke- 31 (1 7 11 3 5 8 9 4 6 2 10) : Panjang jalur = 87

* Jalur ke- 32 (1 7 11 6 5 8 9 3 4 2 10) : Panjang jalur = 100

* Jalur ke- 33 (1 7 11 2 5 8 9 3 6 4 10) : Panjang jalur = 111

* Jalur ke- 34 (1 7 11 10 5 8 9 3 6 2 4) : Panjang jalur = 99

* Jalur ke- 35 (1 7 11 4 8 5 9 3 6 2 10) : Panjang jalur = 94

* Jalur ke- 36 (1 7 11 4 9 8 5 3 6 2 10) : Panjang jalur = 88

* Jalur ke- 37 (1 7 11 4 3 8 9 5 6 2 10) : Panjang jalur = 89

* Jalur ke- 38 (1 7 11 4 6 8 9 3 5 2 10) : Panjang jalur = 95

* Jalur ke- 39 (1 7 11 4 2 8 9 3 6 5 10) : Panjang jalur = 103

* Jalur ke- 40 (1 7 11 4 10 8 9 3 6 2 5) : Panjang jalur = 107

* Jalur ke- 41 (1 7 11 4 5 9 8 3 6 2 10) : Panjang jalur = 87

* Jalur ke- 42 (1 7 11 4 5 3 9 8 6 2 10) : Panjang jalur = 85

* Jalur ke- 43 (1 7 11 4 5 6 9 3 8 2 10) : Panjang jalur = 97

* Jalur ke- 44 (1 7 11 4 5 2 9 3 6 8 10) : Panjang jalur = 105

* Jalur ke- 45 (1 7 11 4 5 10 9 3 6 2 8) : Panjang jalur = 96

* Jalur ke- 46 (1 7 11 4 5 8 3 9 6 2 10) : Panjang jalur = 92

* Jalur ke- 47 (1 7 11 4 5 8 6 3 9 2 10) : Panjang jalur = 98

* Jalur ke- 48 (1 7 11 4 5 8 2 3 6 9 10) : Panjang jalur = 103

* Jalur ke- 49 (1 7 11 4 5 8 10 3 6 2 9) : Panjang jalur = 106

* Jalur ke- 50 (1 7 11 4 5 8 9 6 3 2 10) : Panjang jalur = 94

* Jalur ke- 51 (1 7 11 4 5 8 9 2 6 3 10) : Panjang jalur = 99

* Jalur ke- 52 (1 7 11 4 5 8 9 10 6 2 3) : Panjang jalur = 94

* Jalur ke- 53 (1 7 11 4 5 8 9 3 2 6 10) : Panjang jalur = 86

* Jalur ke- 54 (1 7 11 4 5 8 9 3 10 2 6) : Panjang jalur = 92

* Jalur ke- 55 (1 7 11 4 5 8 9 3 6 10 2) : Panjang jalur = 88

Terbaik = 85, yaitu jalur ke-42 --- DITERIMA sebagai Optimum

Optimum = 85

Jalur terpendek adalah jalur ke-42 dengan panjang 85, maka jalur tersebut terpilih untuk proses diversifikasi selanjutnya. Jalur tersebut ditambahkan ke dalam *tabu list*. Karena panjang jalur tersebut lebih kecil dari Optimum sebelumnya, maka jalur tersebut dipilih sebagai Optimum yang baru.

Iterasi ke- 3 :

Tabu list :

1 1 7 11 4 5 8 3 9 6 2 10

2 1 7 11 4 5 8 9 3 6 2 10

3 1 7 11 4 5 3 9 8 6 2 10

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke- 1 (7 1 11 4 5 3 9 8 6 2 10) : Panjang jalur = 91

* Jalur ke- 2 (11 7 1 4 5 3 9 8 6 2 10) : Panjang jalur = 86

* Jalur ke- 3 (4 7 11 1 5 3 9 8 6 2 10) : Panjang jalur = 109

* Jalur ke- 4 (5 7 11 4 1 3 9 8 6 2 10) : Panjang jalur = 104

* Jalur ke- 5 (3 7 11 4 5 1 9 8 6 2 10) : Panjang jalur = 104

* Jalur ke- 6 (9 7 11 4 5 3 1 8 6 2 10) : Panjang jalur = 100

* Jalur ke- 7 (8 7 11 4 5 3 9 1 6 2 10) : Panjang jalur = 94

* Jalur ke- 8 (6 7 11 4 5 3 9 8 1 2 10) : Panjang jalur = 89

* Jalur ke- 9 (2 7 11 4 5 3 9 8 6 1 10) : Panjang jalur = 95

* Jalur ke- 10 (10 7 11 4 5 3 9 8 6 2 1) : Panjang jalur = 88

* Jalur ke- 11 (1 11 7 4 5 3 9 8 6 2 10) : Panjang jalur = 95

* Jalur ke- 12 (1 4 11 7 5 3 9 8 6 2 10) : Panjang jalur = 94

* Jalur ke- 13 (1 5 11 4 7 3 9 8 6 2 10) : Panjang jalur = 105

* Jalur ke- 14 (1 3 11 4 5 7 9 8 6 2 10) : Panjang jalur = 100

* Jalur ke- 15 (1 9 11 4 5 3 7 8 6 2 10) : Panjang jalur = 97

* Jalur ke- 16 (1 8 11 4 5 3 9 7 6 2 10) : Panjang jalur = 94

* Jalur ke- 17 (1 6 11 4 5 3 9 8 7 2 10) : Panjang jalur = 101

* Jalur ke- 18 (1 2 11 4 5 3 9 8 6 7 10) : Panjang jalur = 102

* Jalur ke- 19 (1 10 11 4 5 3 9 8 6 2 7) : Panjang jalur = 90

* Jalur ke- 20 (1 7 4 11 5 3 9 8 6 2 10) : Panjang jalur = 95

* Jalur ke- 21 (1 7 5 4 11 3 9 8 6 2 10) : Panjang jalur = 88

* Jalur ke- 22 (1 7 3 4 5 11 9 8 6 2 10) : Panjang jalur = 94

* Jalur ke- 23 (1 7 9 4 5 3 11 8 6 2 10) : Panjang jalur = 92

* Jalur ke- 24 (1 7 8 4 5 3 9 11 6 2 10) : Panjang jalur = 95

* Jalur ke- 25 (1 7 6 4 5 3 9 8 11 2 10) : Panjang jalur = 96

* Jalur ke- 26 (1 7 2 4 5 3 9 8 6 11 10) : Panjang jalur = 102

* Jalur ke- 27 (1 7 10 4 5 3 9 8 6 2 11) : Panjang jalur = 106

* Jalur ke- 28 (1 7 11 5 4 3 9 8 6 2 10) : Panjang jalur = 90

* Jalur ke- 29 (1 7 11 3 5 4 9 8 6 2 10) : Panjang jalur = 85

* Jalur ke- 30 (1 7 11 9 5 3 4 8 6 2 10) : Panjang jalur = 92

* Jalur ke- 31 (1 7 11 8 5 3 9 4 6 2 10) : Panjang jalur = 88

* Jalur ke- 32 (1 7 11 6 5 3 9 8 4 2 10) : Panjang jalur = 101

* Jalur ke- 33 (1 7 11 2 5 3 9 8 6 4 10) : Panjang jalur = 110

* Jalur ke- 34 (1 7 11 10 5 3 9 8 6 2 4) : Panjang jalur = 98

* Jalur ke- 35 (1 7 11 4 3 5 9 8 6 2 10) : Panjang jalur = 88

* Jalur ke- 36 (1 7 11 4 9 3 5 8 6 2 10) : Panjang jalur = 91

* Jalur ke- 37 (1 7 11 4 8 3 9 5 6 2 10) : Panjang jalur = 94

* Jalur ke- 38 (1 7 11 4 6 3 9 8 5 2 10) : Panjang jalur = 96

* Jalur ke- 39 (1 7 11 4 2 3 9 8 6 5 10) : Panjang jalur = 103

* Jalur ke- 40 (1 7 11 4 10 3 9 8 6 2 5) : Panjang jalur = 112

* Jalur ke- 41 (1 7 11 4 5 9 3 8 6 2 10) : Panjang jalur = 90

* Jalur ke- 42 (1 7 11 4 5 8 9 3 6 2 10) : Panjang jalur = 86

* Jalur ke- 43 (1 7 11 4 5 6 9 8 3 2 10) : Panjang jalur = 94

* Jalur ke- 44 (1 7 11 4 5 2 9 8 6 3 10) : Panjang jalur = 107

* Jalur ke- 45 (1 7 11 4 5 10 9 8 6 2 3) : Panjang jalur = 101

* Jalur ke- 46 (1 7 11 4 5 3 8 9 6 2 10) : Panjang jalur = 88

* Jalur ke- 47 (1 7 11 4 5 3 6 8 9 2 10) : Panjang jalur = 94

* Jalur ke- 48 (1 7 11 4 5 3 2 8 6 9 10) : Panjang jalur = 102

* Jalur ke- 49 (1 7 11 4 5 3 10 8 6 2 9) : Panjang jalur = 105

* Jalur ke- 50 (1 7 11 4 5 3 9 6 8 2 10) : Panjang jalur = 96

* Jalur ke- 51 (1 7 11 4 5 3 9 2 6 8 10) : Panjang jalur = 96

* Jalur ke- 52 (1 7 11 4 5 3 9 10 6 2 8) : Panjang jalur = 88

* Jalur ke- 53 (1 7 11 4 5 3 9 8 2 6 10) : Panjang jalur = 85

* Jalur ke- 54 (1 7 11 4 5 3 9 8 10 2 6) : Panjang jalur = 86

* Jalur ke- 55 (1 7 11 4 5 3 9 8 6 10 2) : Panjang jalur = 87

Terbaik = 85, yaitu jalur ke-29 --- TIDAK DITERIMA sebagai Optimum

Optimum = 85

Jalur terpendek adalah jalur ke-29 dengan panjang 85, maka jalur tersebut terpilih untuk proses diversifikasi selanjutnya. Jalur tersebut ditambahkan ke dalam *tabu list*. Karena panjang jalur tersebut sama dengan Optimum sebelumnya, maka jalur tersebut tidak diterima sebagai Optimum yang baru. Jadi, Optimum tetap bernilai 85.

Proses diversifikasi berlanjut hingga iterasi berikutnya. Pada iterasi ke-9, diperoleh Optimum baru.

Iterasi ke- 9 :

Tabu list :

1 1 7 11 4 5 8 3 9 6 2 10

2 1 7 11 4 5 8 9 3 6 2 10

3 1 7 11 4 5 3 9 8 6 2 10

4 1 7 11 3 5 4 9 8 6 2 10

5 1 7 11 3 5 4 9 8 2 6 10

6 1 7 11 4 5 3 9 8 2 6 10

7 11 7 1 4 5 3 9 8 2 6 10

8 11 7 1 4 5 3 9 8 6 2 10

9 11 7 1 8 5 3 9 4 6 2 10

TETANGGA (Jalur alternatif berikutnya):

* Jalur ke- 1 (7 11 1 8 5 3 9 4 6 2 10) : Panjang jalur = 94

* Jalur ke- 2 (1 7 11 8 5 3 9 4 6 2 10) : Panjang jalur = 88

* Jalur ke- 3 (8 7 1 11 5 3 9 4 6 2 10) : Panjang jalur = 104

* Jalur ke- 4 (5 7 1 8 11 3 9 4 6 2 10) : Panjang jalur = 92

* Jalur ke- 5 (3 7 1 8 5 11 9 4 6 2 10) : Panjang jalur = 103

* Jalur ke- 6 (9 7 1 8 5 3 11 4 6 2 10) : Panjang jalur = 96

* Jalur ke- 7 (4 7 1 8 5 3 9 11 6 2 10) : Panjang jalur = 102

* Jalur ke- 8 (6 7 1 8 5 3 9 4 11 2 10) : Panjang jalur = 99

* Jalur ke- 9 (2 7 1 8 5 3 9 4 6 11 10) : Panjang jalur = 99

* Jalur ke- 10 (10 7 1 8 5 3 9 4 6 2 11) : Panjang jalur = 92

* Jalur ke- 11 (11 1 7 8 5 3 9 4 6 2 10) : Panjang jalur = 96

* Jalur ke- 12 (11 8 1 7 5 3 9 4 6 2 10) : Panjang jalur = 85

* Jalur ke- 13 (11 5 1 8 7 3 9 4 6 2 10) : Panjang jalur = 101

* Jalur ke- 14 (11 3 1 8 5 7 9 4 6 2 10) : Panjang jalur = 100

* Jalur ke- 15 (11 9 1 8 5 3 7 4 6 2 10) : Panjang jalur = 95

* Jalur ke- 16 (11 4 1 8 5 3 9 7 6 2 10) : Panjang jalur = 100

* Jalur ke- 17 (11 6 1 8 5 3 9 4 7 2 10) : Panjang jalur = 103

* Jalur ke- 18 (11 2 1 8 5 3 9 4 6 7 10) : Panjang jalur = 102

* Jalur ke- 19 (11 10 1 8 5 3 9 4 6 2 7) : Panjang jalur = 93

* Jalur ke- 20 (11 7 8 1 5 3 9 4 6 2 10) : Panjang jalur = 91

* Jalur ke- 21 (11 7 5 8 1 3 9 4 6 2 10) : Panjang jalur = 94

* Jalur ke- 22 (11 7 3 8 5 1 9 4 6 2 10) : Panjang jalur = 101

* Jalur ke- 23 (11 7 9 8 5 3 1 4 6 2 10) : Panjang jalur = 100

* Jalur ke- 24 (11 7 4 8 5 3 9 1 6 2 10) : Panjang jalur = 96

* Jalur ke- 25 (11 7 6 8 5 3 9 4 1 2 10) : Panjang jalur = 102

* Jalur ke- 26 (11 7 2 8 5 3 9 4 6 1 10) : Panjang jalur = 101

* Jalur ke- 27 (11 7 10 8 5 3 9 4 6 2 1) : Panjang jalur = 101

* Jalur ke- 28 (11 7 1 5 8 3 9 4 6 2 10) : Panjang jalur = 93

* Jalur ke- 29 (11 7 1 3 5 8 9 4 6 2 10) : Panjang jalur = 90

* Jalur ke- 30 (11 7 1 9 5 3 8 4 6 2 10) : Panjang jalur = 93

* Jalur ke- 31 (11 7 1 4 5 3 9 8 6 2 10) : Panjang jalur = 86

* Jalur ke- 32 (11 7 1 6 5 3 9 4 8 2 10) : Panjang jalur = 95

* Jalur ke- 33 (11 7 1 2 5 3 9 4 6 8 10) : Panjang jalur = 100

* Jalur ke- 34 (11 7 1 10 5 3 9 4 6 2 8) : Panjang jalur = 95

* Jalur ke- 35 (11 7 1 8 3 5 9 4 6 2 10) : Panjang jalur = 86

* Jalur ke- 36 (11 7 1 8 9 3 5 4 6 2 10) : Panjang jalur = 79

* Jalur ke- 37 (11 7 1 8 4 3 9 5 6 2 10) : Panjang jalur = 88

* Jalur ke- 38 (11 7 1 8 6 3 9 4 5 2 10) : Panjang jalur = 92

* Jalur ke- 39 (11 7 1 8 2 3 9 4 6 5 10) : Panjang jalur = 98

* Jalur ke- 40 (11 7 1 8 10 3 9 4 6 2 5) : Panjang jalur = 106

* Jalur ke- 41 (11 7 1 8 5 9 3 4 6 2 10) : Panjang jalur = 86

* Jalur ke- 42 (11 7 1 8 5 4 9 3 6 2 10) : Panjang jalur = 84

* Jalur ke- 43 (11 7 1 8 5 6 9 4 3 2 10) : Panjang jalur = 94

* Jalur ke- 44 (11 7 1 8 5 2 9 4 6 3 10) : Panjang jalur = 107

* Jalur ke- 45 (11 7 1 8 5 10 9 4 6 2 3) : Panjang jalur = 98

* Jalur ke- 46 (11 7 1 8 5 3 4 9 6 2 10) : Panjang jalur = 88

* Jalur ke- 47 (11 7 1 8 5 3 6 4 9 2 10) : Panjang jalur = 94

* Jalur ke- 48 (11 7 1 8 5 3 2 4 6 9 10) : Panjang jalur = 101

* Jalur ke- 49 (11 7 1 8 5 3 10 4 6 2 9) : Panjang jalur = 107

* Jalur ke- 50 (11 7 1 8 5 3 9 6 4 2 10) : Panjang jalur = 95

* Jalur ke- 51 (11 7 1 8 5 3 9 2 6 4 10) : Panjang jalur = 99

* Jalur ke- 52 (11 7 1 8 5 3 9 10 6 2 4) : Panjang jalur = 93

* Jalur ke- 53 (11 7 1 8 5 3 9 4 2 6 10) : Panjang jalur = 88

* Jalur ke- 54 (11 7 1 8 5 3 9 4 10 2 6) : Panjang jalur = 100

* Jalur ke- 55 (11 7 1 8 5 3 9 4 6 10 2) : Panjang jalur = 94

Terbaik = 79, yaitu jalur ke-36 --- DITERIMA sebagai Optimum

Optimum = 79

Jalur terpendek adalah jalur ke-36 dengan panjang 79, maka jalur tersebut terpilih untuk proses diversifikasi selanjutnya. Jalur tersebut ditambahkan ke dalam *tabu list*. Karena panjang jalur tersebut lebih kecil dari Optimum sebelumnya, maka jalur tersebut dipilih sebagai Optimum yang baru. Proses pencarian berlanjut hingga kriteria pemberhentian dipenuhi, yaitu maksimal iterasi sampai dengan 11.

4.2 Pembahasan

Pada permasalahan di atas, rute terpendek yang diperoleh dari perhitungan menggunakan algoritma TS adalah 79. Hal ini berarti, total jarak terpendek yang ditempuh dalam proses distribusi adalah 79 Km dengan rute perjalanan adalah 11 ⍰ 7 ⍰ 1 ⍰ 8 ⍰ 9 ⍰ 3 ⍰ 5 ⍰ 4 ⍰ 6 ⍰ 2 ⍰ 10 (Computa ï ALNEC ï IT COM ï WOW ï WKM ï Dian Kencana ï Saintech ï Fajar Aircond ï Surya I ï Rifani ï Larisa ï Computa). Karena rute yang terbentuk merupakan sebuah *cycle*, maka apabila rute diawali dari depot maka rute menjadi 1 ⍰ 8 ⍰ 9 ⍰ 3 ⍰ 5 ⍰ 4 ⍰ 6 ⍰ 2 ⍰ 10 ⍰ 11 ⍰ 7 (IT COM ï WOW ï WKM ï Dian Kencana ï Saintech ï Fajar Aircond ï Surya I ï Rifani ï Larisa ï Computa ï ALNEC) dan rute akan berakhir pada depot.

Adakalanya rute yang ditemukan mempunyai panjang yang sama sehingga terdapat lebih dari satu solusi. Solusi terbaik yang diambil adalah solusi yang

pertama kali ditemukan. Tapi, apabila ditemukan solusi terbaik yang mempunyai urutan berbeda namun panjang rute sama, maka rute tersebut tetap dipilih sebagai rute yang dipilih untuk proses diversifikasi selanjutnya.

Dengan membandingkan metode yang digunakan oleh IT COMM dan algoritma *Tabu search* diperoleh hasil yang berbeda. Perhitungan menggunakan algoritma *Tabu search* memberikan hasil yang lebih optimal atau dengan kata lain didapatkan jarak yang lebih minimal. Hal ini berarti algoritma *Tabu search* merupakan salah satu algoritma yang cukup efektif untuk menyelesaikan VRP.

Proses perhitungan secara manual membutuhkan waktu yang lama. Hal ini dikarenakan banyak iterasi sama dengan banyak *dealer* yaitu 11 dan tiap iterasi terdapat 55 solusi alternatif. Oleh sebab itu, penulis membangun program menggunakan *software* Delphi untuk mempermudah perhitungan.

Dengan menggunakan program Delphi, pencarian rute paling optimal dari 11 *dealer* dengan maksimum iterasi 11 hanya membutuhkan waktu 4 detik. Melihat lama waktu yang digunakan untuk perhitungan menggunakan algoritma *Tabu search*, dapat dikatakan bahwa penggunaan program Delphi jauh lebih cepat dan akurat.

Namun terdapat kelemahan dari program yang dibuat dalam bahasa Delphi di atas, yaitu program statis. *Input* jarak antar *dealer* dilakukan di dalam kode program, akibatnya apabila terdapat penambahan jumlah *dealer* maka *input* jarak tidak dapat secara otomatis dilakukan. *Input* jarak dilakukan dengan merubah kode di dalam program.

BAB 5
PENUTUP

5.1 Simpulan

Dari penelitian yang telah dilakukan, dapat diambil simpulan mengenai kinerja pencarian rute perjalanan kendaraan optimal menggunakan algoritma *tabu search* yaitu proses perhitungan menggunakan algoritma *tabu search* terdiri dari 6 langkah. Langkah pertama yaitu menentukan solusi awal pada iterasi 0 dan menetapkan nilai solusi awal sebagai nilai solusi optimum. Langkah kedua yaitu mencari solusi-solusi alternatif yang tidak melanggar kriteria tabu. Langkah ke tiga yaitu memilih solusi terbaik diantara solusi alternatif pada langkah ke dua. Langkah ke empat yaitu memilih nilai solusi optimum. Apabila nilai solusi terbaik pada langkah ke tiga lebih kecil dari nilai solusi optimum awal, maka solusi terbaik dipilih sebagai solusi optimum baru. Langkah ke lima yaitu memperbarui *tabu list* dengan memasukkan solusi optimum baru. Langkah ke enam yaitu apabila kriteria pemberhentian dipenuhi maka proses perhitungan berhenti dan diperoleh solusi optimum, jika tidak proses kembali berulang dimulai dari langkah ke dua.

Pada permasalahan di atas, total jarak terpendek yang ditempuh dalam proses distribusi adalah 79 Km dengan rute perjalanan IT COM ï WOW ï WKM ï Dian Kencana ï Saintech ï Fajar Aircond ï Surya I ï Rifani ï Larisa ï Computa ï ALNEC dan rute akan berakhir pada depot.

5.2 Saran

(1) Berdasarkan pembahasan di atas, disarankan kepada Perusahaan IT COMM untuk menggunakan metode algoritma *Tabu search* dalam proses distribusi sehingga biaya yang dikeluarkan minimal.

(2) Pada skripsi ini, program yang dibangun statis. *Input* jarak antar dealer dilakukan di dalam kode program. Untuk itu perlu diadakan penelitian lebih lanjut agar *input* jarak dapat dilakukan tanpa merubah kode di dalam program.

(3) Perlu diadakan penelitian lebih lanjut untuk memperlihatkan dan membuktikan keefektifan, kelebihan, keakuratan dan kelemahan dari algoritma *Tabu search* (TS), dengan tujuan untuk membandingkan seluruh algoritma heuristik yang ada pada berbagai data dengan jumlah titik yang lebih banyak dari yang saat ini diteliti.

DAFTAR PUSTAKA

Berlianty, I dan Miftahol, A. 2010. *Teknik-Teknik Optimasi Heuristik*. Yogyakarta: Graha Ilmu.

Gendreau, M,. and Potvin, J.Y (eds). 2010. *Handbook of Metaheuristics: Second Edition*. New York: Springer Science+Business Media.

Glover, F and Kochenberger, G.A (eds). 2003. *Handbook of Metaheuristics*. Dordrecht: Kluwer Academic Publisher.

Glover, F and Laguna, M. 1997. *Tabu Search*. Massachusetts: Kluwer Academic Publisher.

Hertz, A., Taillard, E,. and de Werra, D. 2002. *A Tutorial on Tabu Search*. EPFL, Departement de Mathematiques, MA-Ecublens, CH-1015: Laussanne. http://www.cs.colostate.edu/~whitley/CS640/hertz92tutorial.pdf. Diakses tanggal 09 Februari 2011 Pukul 19:41.

Poetra, F.H. 2010. *Aplikasi Algoritma Tabu Search Pada Pencarian Jalur Terpendek*. Skripsi FMIPA Universitas Sumatera Utara: Medan.

Satria, W., Siallagan, M.P., dan Novani, S. 2004. *Penerapan Metode Algoritma Genetik untuk Memecahkan Masalah Penentuan rute Kendaraan Berkendala Kapasitas*. Universitas Komputer Indonesia.

Sutarno, H., Priatna, N., Nurjanah. 2003. *Common TextBook Matematika Diskrit*.

Lampiran 1

List Program Form Utama

```
unit Unit1;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, ExtCtrls, jpeg;
type
  TForm1 = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    Program1: TMenuItem;
    abuSearch1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    utorial1: TMenuItem;
    Timer1: TTimer;
    Utama1: TMenuItem;
    Image1: TImage;
    procedure Exit1Click(Sender: TObject);
    procedure abuSearch1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure utorial1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
uses Unit2, Unit3, Unit4;
{$R *.dfm}
procedure TForm1.Exit1Click(Sender: TObject);
begin
        if messageDlg ('Anda yakin akan keluar?',mtconfirmation,mbOKCancel,0)
                =mrOk then
                application.Terminate;
        end;
procedure TForm1.abuSearch1Click(Sender: TObject);
begin
        form1.Hide;
```

```
        form2.show;
end;

procedure TForm1.About1Click(Sender: TObject);
begin
  MessageDlg('Skin Form' + #13#10 +
        'Ditulis oleh Fajar Eska Pradhana' + #13#10#13#10 +
        'Registered : to Public'+ #13#10#13#10 +
        'E-Mail : fajarpradhana@gmail.com' + #13#10 +
        'Facebook : fat_jar@yahoo.com' + #13#10#13#10 +
        'Copyright © Faj Software - Delphi Source Code.',
        mtInformation,[MbOk],0);
end;

procedure TForm1.utorial1Click(Sender: TObject);
begin
        form1.Hide;
        form4.show;
end;

procedure TForm1.FormCreate(Sender: TObject);
var MyBitmap:TBitmap;
begin
        MyBitmap:=Tbitmap.Create;
        MyBitmap.LoadFromFile('e.bmp');
        form1.Brush.Bitmap:=MyBitmap;
        animatewindow(self.Handle,1500,AW_CENTER or AW_ACTIVATE);
        timer1.interval:=500;
        timer1.enabled:=true;
end;
end.
```

Lampiran 2

***List* program form Tabu Search**

```
unit Unit2;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Menus, StdCtrls, XPMan, ExtCtrls;
type
  TForm2 = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    Program1: TMenuItem;
    abuSearch1: TMenuItem;
    Help1: TMenuItem;
    About1: TMenuItem;
    utorial1: TMenuItem;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    Edit10: TEdit;
    XPManifest1: TXPManifest;
    SaveDialog1: TSaveDialog;
    Action: TGroupBox;
    hitung: TButton;
    simpan: TButton;
    bersihkan: TButton;
    Utama1: TMenuItem;
    memo1: TMemo;
    Label11: TLabel;
    Edit11: TEdit;
    Timer1: TTimer;
    function jalur(index:integer):real;
    procedure IsiTabel;
    procedure cekawal;
    procedure tampung;
    procedure kembalikan;
    procedure newtabulist;
    procedure hanyaangka(var Key: Char);
    procedure inputrute(var Key: Char);
    procedure Exit1Click(Sender: TObject);
    procedure About1Click(Sender: TObject);
    procedure utorial1Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit2Change(Sender: TObject);
    procedure Edit3Change(Sender: TObject);
    procedure Edit4Change(Sender: TObject);
    procedure Edit5Change(Sender: TObject);
    procedure Edit6Change(Sender: TObject);
    procedure Edit7Change(Sender: TObject);
    procedure Edit8Change(Sender: TObject);
    procedure Edit9Change(Sender: TObject);
```

```
  procedure  Edit10Change(Sender: TObject);
  procedure  Edit11Change(Sender: TObject);
  procedure  Edit1KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit2KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit3KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit4KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit5KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit6KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit7KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit8KeyPress(Sender: TObject; var Key: Char);
  procedure  Edit9KeyPress(Sender: TObject; var Key: Char);
  procedure Edit10KeyPress(Sender: TObject; var Key: Char);
  procedure Edit11KeyPress(Sender: TObject; var Key: Char);
  procedure  Utama1Click(Sender: TObject);
  procedure  FormCreate(Sender: TObject);
  procedure  hitungClick(Sender: TObject);
  procedure  simpanClick(Sender: TObject);
  procedure  bersihkanClick(Sender: TObject);

 private
   { Private declarations }
 public
   { Public declarations }
 end;

var
  Form2: TForm2;
  Tabel:Array[1..11,1..11] of real;

dealer,dealert,dealernext:array[1..11] of integer;

i,j,k,a,c,d,b,n,z,e,f,temp,urutan,stat:integer;
  jalurr,jalurb,jalurgm:string;

Optimum,terbaik{,cost},tempuh:real;
  tblist: array [1..11] of String;

implementation

uses Unit3, Unit4, Unit1;

{$R *.dfm}
procedure  Tform2.hanyaangka(var Key: Char);
begin
if not (((key>='0') and (key<='9') or (key=#8))) then
  key:=#0;
end;

procedure Tform2.inputrute(var Key: Char);
begin
if not (((key>='0') and (key<='9') or (key=#8))) then
  key:=#0;
end;

procedure  Tform2.tampung; //menampung data dealer.
begin
for j:=1 to 11 do
  begin
    dealert[j]:=dealer[j];
  end;
end;

procedure  Tform2.kembalikan;// mengembalikan dealer yang sudah ditukar2 dengan nilai awal.
```

```
begin
for j:=1 to 11 do
  begin
    dealer[j]:=dealert[j];
  end;
end;

procedure  Tform2.newtabulist;//tabu
list  baru,  dijalankan  ketika
ditemukan nilai minimal(best so far)
baru. untuk dijadikan tabu list
berikutnya.
begin
  for c:=1 to 11 do
    begin
      dealer[c]:=dealernext[c];
    end;
end;

function
Tform2.jalur(index:integer):real;//Sw
ap  jalur.  1234567891011-
>2134567891011->dst
//var
//  TEMPUH:REAL;
begin
  tempuh:=0;
  //Swap jalur  berdasarkan indeks...
  if index = 1 then
    begin
      temp:=dealer[1];
      dealer[1]:=dealer[2];
      dealer[2]:=temp;
    end;
  if index = 2 then
    begin
      temp:=dealer[1];
      dealer[1]:=dealer[3];
      dealer[3]:=temp;
    end;
  if index =  3 then
    begin
      temp:=dealer[1];
      dealer[1]:=dealer[4];
      dealer[4]:=temp;
    end;
if index = 4 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[5];
    dealer[5]:=temp;
  end;
if index = 5 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[6];
    dealer[6]:=temp;
  end;
if index = 6 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[7];
    dealer[7]:=temp;
  end;
if index = 7 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[8];
    dealer[8]:=temp;
  end;
if index = 8 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 9 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 10 then
  begin
    temp:=dealer[1];
    dealer[1]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 11 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[3];
```

```
    dealer[3]:=temp;
  end;
if index = 12 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[4];
    dealer[4]:=temp;
  end;
if index = 13 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[5];
    dealer[5]:=temp;
  end;
if index = 14 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[6];
    dealer[6]:=temp;
  end;
if index = 15 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[7];
    dealer[7]:=temp;
  end;
if index = 16 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[8];
    dealer[8]:=temp;
  end;
if index = 17 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 18 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 19 then
  begin
    temp:=dealer[2];
    dealer[2]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 20 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[4];
    dealer[4]:=temp;
  end;
if index = 21 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[5];
    dealer[5]:=temp;
  end;
if index = 22 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[6];
    dealer[6]:=temp;
  end;
if index = 23 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[7];
    dealer[7]:=temp;
  end;
if index = 24 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[8];
    dealer[8]:=temp;
  end;
if index = 25 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 26 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[10];
    dealer[10]:=temp;
  end;
```

```
if index = 27 then
  begin
    temp:=dealer[3];
    dealer[3]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 28 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[5];
    dealer[5]:=temp;
  end;
if index = 29 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[6];
    dealer[6]:=temp;
  end;
if index = 30 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[7];
    dealer[7]:=temp;
  end;
if index = 31 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[8];
    dealer[8]:=temp;
  end;
if index = 32 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 33 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 34 then
  begin
    temp:=dealer[4];
    dealer[4]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 35 then
  begin
    temp:=dealer[5];
    dealer[5]:=dealer[6];
    dealer[6]:=temp;
  end;
if index = 36 then
  begin
    temp:=dealer[5];
    dealer[5]:=dealer[7];
    dealer[7]:=temp;
  end;
if index = 37 then
  begin
    temp:=dealer[5];
    dealer[5]:=dealer[8];
    dealer[8]:=temp;
  end;
if index = 38 then
  begin
    temp:=dealer[5];
    dealer[5]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 39 then
  begin
    temp:=dealer[5];
    dealer[5]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 40 then
  begin
    temp:=dealer[5];
    dealer[5]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 41 then
  begin
    temp:=dealer[6];
    dealer[6]:=dealer[7];
    dealer[7]:=temp;
  end;
if index = 42 then
  begin
```

```
    temp:=dealer[6];
    dealer[6]:=dealer[8];
    dealer[8]:=temp;
  end;
if index = 43 then
  begin
    temp:=dealer[6];
    dealer[6]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 44 then
  begin
    temp:=dealer[6];
    dealer[6]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 45 then
  begin
    temp:=dealer[6];
    dealer[6]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 46 then
  begin
    temp:=dealer[7];
    dealer[7]:=dealer[8];
    dealer[8]:=temp;
  end;
 if index = 47 then
  begin
    temp:=dealer[7];
    dealer[7]:=dealer[9];
    dealer[9]:=temp;
  end;
 if index = 48 then
  begin
    temp:=dealer[7];
    dealer[7]:=dealer[10];
    dealer[10]:=temp;
  end;
  if index = 49 then
  begin
    temp:=dealer[7];
    dealer[7]:=dealer[11];
    dealer[11]:=temp;
  end;
 if index = 50 then
  begin
    temp:=dealer[8];
    dealer[8]:=dealer[9];
    dealer[9]:=temp;
  end;
if index = 51 then
  begin
    temp:=dealer[8];
    dealer[8]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 52 then
  begin
    temp:=dealer[8];
    dealer[8]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 53 then
  begin
    temp:=dealer[9];
    dealer[9]:=dealer[10];
    dealer[10]:=temp;
  end;
if index = 54 then
  begin
    temp:=dealer[9];
    dealer[9]:=dealer[11];
    dealer[11]:=temp;
  end;
if index = 55 then
  begin
    temp:=dealer[10];
    dealer[10]:=dealer[11];
    dealer[11]:=temp;
  end;

jalurr:='';//pengkosongan variable jalurr untuk penyimpanan yang baru.
  for i:=1 to 11 do
    begin
      jalurr:='+jalurr+' '+inttostr(dealer[i])+'';  //menyimpan rute jalur pada masing2 jalur.
    end;
```

```
//menghitung jarak untuk tiap rute.
for i:=1 to 11 do
  begin
    if i<11 then
      begin

tempuh:=tempuh+Tabel[dealer[i],dealer[i+1]];
      end
    else
      begin

tempuh:=tempuh+Tabel[dealer[11],dealer[1]];
      end;
  end;
// ceking,ada jalur yang sama apa tidak?? kalo sama,lewati. jika tidak sama, lanjut...
if (jalurr<>tblist[1]) and (jalurr<>tblist[2]) and (jalurr<>tblist[3]) and
   (jalurr<>tblist[4]) and (jalurr<>tblist[5]) and (jalurr<>tblist[6]) and (jalurr<>tblist[7])
   and (jalurr<>tblist[8]) and (jalurr<>tblist[9]) and (jalurr<>tblist[10])and (jalurr<>tblist[11]) then
  begin
    if tempuh<terbaik then//jika jalur tidak sama dan kurang dari terbaik, maka dia dijadikan terbaik yang baru.
      begin
        terbaik:=tempuh;
        jalurb:=jalurr; //jika terbaik< global min, tampung jalur terbaik untuk dijadikan jalur global min
        for z:=1 to 11 do
          begin
            dealernext[z]:=dealer[z];// karena dia terbaik, maka akan dijadikan tabulist yang selanjutnya.
          end;
        urutan:=index;
      end;
  end;
memo1.Lines.Add('* Jalur ke-'+inttostr(index)+' ('+jalurr+') : Panjang jalur = '+floattostr(tempuh)+");
jalur:=tempuh;
end;

procedure Tform2.IsiTabel;
begin
Tabel[1,1]:=0;
Tabel[1,2]:=10;
Tabel[1,3]:=10;
Tabel[1,4]:=8;
Tabel[1,5]:=9;
Tabel[1,6]:=10;
Tabel[1,7]:=5;
Tabel[1,8]:=5;
Tabel[1,9]:=12;
Tabel[1,10]:=10;
Tabel[1,11]:=12;
Tabel[2,1]:=10;
Tabel[2,2]:=0;
Tabel[2,3]:=10;
Tabel[2,4]:=12;
Tabel[2,5]:=13;
Tabel[2,6]:=10;
Tabel[2,7]:=15;
Tabel[2,8]:=13;
Tabel[2,9]:=18;
Tabel[2,10]:=12;
Tabel[2,11]:=15;
Tabel[3,1]:=10;
Tabel[3,2]:=10;
Tabel[3,3]:=0;
Tabel[3,4]:=4;
Tabel[3,5]:=4;
Tabel[3,6]:=10;
Tabel[3,7]:=10;
Tabel[3,8]:=8;
Tabel[3,9]:=9;
Tabel[3,10]:=16;
Tabel[3,11]:=5;
Tabel[4,1]:=8;
```

```
Tabel[4,2]:=12;
Tabel[4,3]:=4;
Tabel[4,4]:=0;
Tabel[4,5]:=2;
Tabel[4,6]:=9;
Tabel[4,7]:=8;
Tabel[4,8]:=9;
Tabel[4,9]:=9;
Tabel[4,10]:=17;
Tabel[4,11]:=5;
Tabel[5,1]:=9;
Tabel[5,2]:=13;
Tabel[5,3]:=4;
Tabel[5,4]:=2;
Tabel[5,5]:=0;
Tabel[5,6]:=10;
Tabel[5,7]:=8;
Tabel[5,8]:=8;
Tabel[5,9]:=10;
Tabel[5,10]:=14;
Tabel[5,11]:=10;
Tabel[6,1]:=10;
Tabel[6,2]:=10;
Tabel[6,3]:=10;
Tabel[6,4]:=9;
Tabel[6,5]:=10;
Tabel[6,6]:=0;
Tabel[6,7]:=15;
Tabel[6,8]:=13;
Tabel[6,9]:=17;
Tabel[6,10]:=12;
Tabel[6,11]:=15;
Tabel[7,1]:=5;
Tabel[7,2]:=15;
Tabel[7,3]:=10;
Tabel[7,4]:=8;
Tabel[7,5]:=8;
Tabel[7,6]:=15;
Tabel[7,7]:=0;
Tabel[7,8]:=10;
Tabel[7,9]:=16;
Tabel[7,10]:=10;
Tabel[7,11]:=6;
Tabel[8,1]:=5;
Tabel[8,2]:=13;
Tabel[8,3]:=8;
Tabel[8,4]:=9;
Tabel[8,5]:=8;
Tabel[8,6]:=13;
Tabel[8,7]:=10;
Tabel[8,8]:=0;
Tabel[8,9]:=9;
Tabel[8,10]:=14;
Tabel[8,11]:=6;
Tabel[9,1]:=12;
Tabel[9,2]:=18;
Tabel[9,3]:=9;
Tabel[9,4]:=9;
Tabel[9,5]:=10;
Tabel[9,6]:=17;
Tabel[9,7]:=16;
Tabel[9,8]:=9;
Tabel[9,9]:=0;
Tabel[9,10]:=17;
Tabel[9,11]:=9;
Tabel[10,1]:=10;
Tabel[10,2]:=12;
Tabel[10,3]:=16;
Tabel[10,4]:=17;
Tabel[10,5]:=14;
Tabel[10,6]:=12;
Tabel[10,7]:=10;
Tabel[10,8]:=14;
Tabel[10,9]:=17;
Tabel[10,10]:=0;
Tabel[10,11]:=8;
Tabel[11,1]:=12;
Tabel[11,2]:=15;
Tabel[11,3]:=5;
Tabel[11,4]:=5;
Tabel[11,5]:=10;
Tabel[11,6]:=15;
Tabel[11,7]:=6;
Tabel[11,8]:=6;
Tabel[11,9]:=9;
Tabel[11,10]:=8;
Tabel[11,11]:=0;

end;
```

```
procedure
TForm2.Exit1Click(Sender:
TObject);
begin
 if messageDlg ('Anda yakin akan
keluar?',mtconfirmation,mbOKCanc
el,0)
   =mrOk then
 application.Terminate;
end;

procedure
TForm2.About1Click(Sender:
TObject);
begin
 MessageDlg('Skin Form' + #13#10
+
        'Ditulis oleh Fajar Eska
Pradhana' + #13#10#13#10 +
        'Registered : to Public'+
#13#10#13#10 +
        'E-Mail :
fajarpradhana@gmail.com' + #13#10
+
        'Facebook :
fat_jar@yahoo.com' +
#13#10#13#10 +
        'Copyright © Faj Software -
Delphi Source Code.',
        mtInformation,[MbOk],0);
end;

procedure
TForm2.utorial1Click(Sender:
TObject);
begin
form2.Hide;
form4.show;
end;

procedure Tform2.cekawal;//cek
awal solusi.
begin
 if ((edit1.Text=edit2.Text) and
(edit2.Text<>''))or
   ((edit1.Text=edit3.Text) and
(edit3.Text<>'')) or
   ((edit1.Text=edit4.Text) and
(edit4.Text<>'')) or
   ((edit1.Text=edit5.Text) and
(edit5.Text<>'')) or
   ((edit1.Text=edit6.Text) and
(edit6.Text<>'')) or
   ((edit1.Text=edit7.Text) and
(edit7.Text<>'')) or
   ((edit1.Text=edit8.Text) and
(edit8.Text<>'')) or
   ((edit1.Text=edit9.Text) and
(edit9.Text<>'')) or
   ((edit1.Text=edit10.Text) and
(edit10.Text<>'')) or
   ((edit1.Text=edit11.Text) and
(edit11.Text<>'')) or
   ((edit2.Text=edit3.Text) and
(edit3.Text<>'')) or
   ((edit2.Text=edit4.Text) and
(edit4.Text<>'')) or
   ((edit2.Text=edit5.Text) and
(edit5.Text<>'')) or
   ((edit2.Text=edit6.Text) and
(edit6.Text<>'')) or
   ((edit2.Text=edit7.Text) and
(edit7.Text<>'')) or
   ((edit2.Text=edit8.Text) and
(edit8.Text<>'')) or
   ((edit2.Text=edit9.Text) and
(edit9.Text<>'')) or
   ((edit2.Text=edit10.Text) and
(edit10.Text<>'')) or
   ((edit2.Text=edit11.Text) and
(edit11.Text<>''))or
   ((edit3.Text=edit4.Text) and
(edit4.Text<>'')) or
   ((edit3.Text=edit5.Text) and
(edit5.Text<>'')) or
   ((edit3.Text=edit6.Text) and
(edit6.Text<>'')) or
   ((edit3.Text=edit7.Text) and
(edit7.Text<>'')) or
   ((edit3.Text=edit8.Text) and
(edit8.Text<>'')) or
```

```
    ((edit3.Text=edit9.Text) and
(edit9.Text<>")) or
    ((edit3.Text=edit10.Text) and
(edit10.Text<>")) or
    ((edit3.Text=edit11.Text) and
(edit11.Text<>"))or
    ((edit4.Text=edit5.Text) and
(edit5.Text<>")) or
    ((edit4.Text=edit6.Text) and
(edit6.Text<>")) or
    ((edit4.Text=edit7.Text) and
(edit7.Text<>")) or
    ((edit4.Text=edit9.Text) and
(edit8.Text<>")) or
    ((edit4.Text=edit10.Text) and
(edit10.Text<>")) or
    ((edit4.Text=edit11.Text) and
(edit11.Text<>")) or
    ((edit5.Text=edit6.Text) and
(edit6.Text<>")) or
    ((edit5.Text=edit8.Text) and
(edit7.Text<>")) or
    ((edit5.Text=edit8.Text) and
(edit8.Text<>")) or
    ((edit5.Text=edit9.Text) and
(edit9.Text<>")) or
    ((edit5.Text=edit10.Text) and
(edit10.Text<>")) or
    ((edit5.Text=edit11.Text) and
(edit11.Text<>")) or
    ((edit6.Text=edit7.Text) and
(edit7.Text<>")) or
    ((edit6.Text=edit8.Text) and
(edit8.Text<>")) or
    ((edit6.Text=edit9.Text) and
(edit9.Text<>")) or
    ((edit6.Text=edit10.Text) and
(edit10.Text<>")) or
    ((edit6.Text=edit11.Text) and
(edit11.Text<>")) or
    ((edit7.Text=edit8.Text) and
(edit8.Text<>")) or
    ((edit7.Text=edit9.Text) and
(edit9.Text<>")) or
    ((edit7.Text=edit10.Text) and
(edit10.Text<>")) or
    ((edit7.Text=edit11.Text) and
(edit11.Text<>")) or
    ((edit8.Text=edit9.Text) and
(edit9.Text<>")) or
    ((edit8.Text=edit10.Text) and
(edit10.Text<>")) or
    ((edit8.Text=edit11.Text) and
(edit11.Text<>")) or
    ((edit9.Text=edit10.Text) and
(edit10.Text<>"))or
    ((edit9.Text=edit11.Text) and
(edit11.Text<>")) or
    ((edit10.Text=edit11.Text) and
(edit11.Text<>")) then
   begin
    showmessage('indeks dealer
yang sama terdeteksi, periksa
kembali dan pastikan tidak ada
pengulangan indeks.');
    hitung.enabled:=false;
    exit;
    end
  else
   begin
    if (edit1.Text<>") and
(edit2.Text<>") and (edit3.Text<>")
and (edit4.Text<>") and
(edit5.Text<>") and (edit6.Text<>")
and (edit7.Text<>")
    and (edit8.Text<>") and
(edit9.Text<>") and (edit7.Text<>")
and (edit8.Text<>")
    and (edit9.Text<>") and
(edit10.Text<>")and
(edit11.Text<>")then
     begin
      hitung.Enabled:=true;
     end
    else
     begin
      hitung.Enabled:=false;
     end;
   end;
end;
```

```
procedure
TForm2.Edit1Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit2Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit3Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit4Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit5Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit6Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit7Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit8Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit9Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit10Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit11Change(Sender:
TObject);
begin
cekawal;
end;

procedure
TForm2.Edit1KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit2KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;
```

```
procedure
TForm2.Edit3KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit4KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit5KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit6KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit7KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit8KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit9KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit10KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Edit11KeyPress(Sender:
TObject; var Key: Char);
begin
inputrute(Key);
end;

procedure
TForm2.Utama1Click(Sender:
TObject);
begin
form2.Hide;
form1.show;
end;

procedure
TForm2.FormCreate(Sender:
TObject);
begin
hitung.Enabled:=false;
simpan.Enabled:=false;
bersihkan.Enabled:=false;
end;

procedure
TForm2.hitungClick(Sender:
TObject);
var waktu1,waktu2,selisih:Ttime;

begin
waktu1:=time;
timer1.Enabled:=true;
```

```
simpan.Enabled:=true;
bersihkan.Enabled:=true;
IsiTabel; //-> Memanggil Prosedur IsiTabel.
    for i:=1 to 11 do
      begin
        tblist[i]:='';
      end;
      jalurgm:='';
      tempuh:=0;
     //definisi rute awal.
     dealer[1]:=strtoint(edit1.Text);
     dealer[2]:=strtoint(edit2.Text);
     dealer[3]:=strtoint(edit3.Text);
     dealer[4]:=strtoint(edit4.Text);
     dealer[5]:=strtoint(edit5.Text);
     dealer[6]:=strtoint(edit6.Text);
     dealer[7]:=strtoint(edit7.Text);
     dealer[8]:=strtoint(edit8.Text);
     dealer[9]:=strtoint(edit9.Text);
     dealer[10]:=strtoint(edit10.Text);
     dealer[11]:=strtoint(edit11.Text);

     //Rute awal sebagi tabulist pertama.
     for i:=1 to 11 do
       begin
         tblist[1]:="+tblist[1]+' '+inttostr(dealer[i])+";
       end;

     //Kalkulasi jarak tabulist pertama.
     for i:=1 to 11 do
       begin
         if i<11 then
           begin
             tempuh:=tempuh+Tabel[dealer[i],dealer[i+1]];
           end
         else
           begin
             tempuh:=tempuh+Tabel[dealer[11],dealer[1]];
           end;
       end;


     terbaik:=100000;///maksimalkan nilai terbaik
     optimum:=tempuh;//menjadikan tempuh pertama sebagai optimum
     jalurgm:=tblist[1];//menjadikan tabulist pertama sebagai jalur optimum
     if stat=1 then
```

```
        begin
          memo1.Lines.Add('');
          memo1.Lines.Add('');

memo1.Lines.Add('==========================================
=');
          memo1.Lines.Add('');
          memo1.Lines.Add('');
        end;
      memo1.Lines.Add('Algoritma T A B U   S E A R C H');
      memo1.Lines.Add('----------------------');
      memo1.Lines.Add('');
      memo1.Lines.Add('');
      memo1.Lines.Add('');
      memo1.Lines.Add('Jarak Antar Dealer : ');
      memo1.Lines.Add('     1        2        3        4        5        6        7        8
9        10        11');
      for i:=1 to 11 do
        begin
          memo1.Lines.Add(''+inttostr(i)+'          |   '+floattostr(tabel[i,1])+'          |
'+floattostr(tabel[i,2])+'      |  '+floattostr(tabel[i,3])+'      |  '+floattostr(tabel[i,4])+'
|  '+floattostr(tabel[i,5])+'     |  '+floattostr(tabel[i,6])+'     |  '+floattostr(tabel[i,7])+'
|      '+floattostr(tabel[i,8])+'                |      '+floattostr(tabel[i,9])+'              |
'+floattostr(tabel[i,10])+'      |  '+floattostr(tabel[i,10])+'');
        end;
      memo1.Lines.Add('');
      memo1.Lines.Add('');
      memo1.Lines.Add('Iterasi ke- 1 :');
      memo1.Lines.Add('----------------');
      memo1.Lines.Add('Tabu List 1 : '+tblist[1]+' Panjang Jalur =
'+floattostr(tempuh)+'');
      memo1.Lines.Add('TETANGGA (Jalur alternatif berikutnya):');

      //bagian untuk menukar2 posisi dealer. tukar dilakukan pada fungsi jalur().
      for k:=1 to 55 do
        begin
          tampung;//prosedur untuk menampung nilai sebelum ditukar.
          jalur(k);//melakukan penukaran sesuai indeks.
          kembalikan;//melakukan pengembalian nilai. menjadi seperti awal. untuk
ditukar lagi.
        end;

      //mengisi jalurr dengan tabulist selanjutnya.
      for b:=1 to 11 do
      begin
          jalurr:=''+jalurr+' '+inttostr(dealernext[b])+'';
```

```
      end;

    //jika terbaik< optimum maka akan diterima sebagai optimum. jika tidak, di abaikan.
    if terbaik<optimum then
      begin
        jalurgm:='';
        jalurgm:=jalurb;//set jalurgm menjadi jalurrb jika terbaik < optimum
        optimum:=terbaik;
        memo1.Lines.Add('Terbaik = '+floattostr(terbaik)+', yaitu jalur ke-'+inttostr(urutan)+' --- DITERIMA sebagai Optimum');
      end
    else
      begin
        memo1.Lines.Add('Terbaik = '+floattostr(terbaik)+', yaitu jalur ke-'+inttostr(urutan)+' --- TIDAK DITERIMA sebagai GlobalMin');
      end;

    memo1.Lines.Add('Globalmin = '+floattostr(optimum)+');memo1.Lines.Add('');memo1.Lines.Add('');

    //pengulangan untuk iterasi selanjutnya (2-10)
    for d:=1 to 10 do
      begin
        memo1.Lines.Add('Iterasi ke- '+inttostr(d+1)+' :');
        memo1.Lines.Add('----------------');memo1.Lines.Add('');

        terbaik:=100000;//memaksimalkan terbaik pada tiap iterasi
        jalurr:='';

        //ambil list dealer untuk dijadikan tabu list selanjutnya(tabulist ke d+1)
        newtabulist;
        for i:=1 to 11 do
          begin
            tblist[d+1]:='+tblist[d+1]+' '+inttostr(dealer[i])+'';
          end;

        //cetak daftar tabu list yang dulu hingga kini....
        memo1.Lines.Add('Tabu List : ');
        for n:=1 to d+1 do
          begin
            memo1.Lines.Add('+inttostr(n)+'     '+tblist[n]+'');
          end;
        memo1.Lines.Add('');

        //jalur berikutnya..
```

```
          memo1.Lines.Add('TETANGGA (Jalur alternatif berikutnya):');
          for k:=1 to 55 do
            begin
              tampung;
              jalur(k);
              kembalikan;
            end;
          for b:=1 to 11 do
            begin
              jalurr:="+jalurr+' '+inttostr(dealernext[b])+";
            end;
          if terbaik<optimum then
            begin
              jalurgm:=";
              jalurgm:=jalurb;//jika terbaik < optimum, set jalurgm menjadi jalurrb
              optimum:=terbaik;
              memo1.Lines.Add('Terbaik = '+floattostr(terbaik)+', yaitu jalur ke-
'+inttostr(urutan)+' --- DITERIMA sebagai Optimum');
            end
          else
            begin
              memo1.Lines.Add('Terbaik = '+floattostr(terbaik)+', yaitu jalur ke-
'+inttostr(urutan)+' --- TIDAK DITERIMA sebagai Optimum');
            end;
          memo1.Lines.Add('Optimum = '+floattostr(optimum)+");
          memo1.Lines.Add(");memo1.Lines.Add(");

        end;
        memo1.Lines.Add('Rute terpendek adalah : '+jalurgm+' dengan jarak =
'+floattostr(optimum)+"); //cetak hasil akhir... :)
        memo1.Lines.Add(");
        memo1.Lines.Add(");
        memo1.Lines.Add('.::----------Proses Pencarian Solusi SELESAI-------
----------::.');
        stat:=1;
    waktu2:=time;
    selisih:=(waktu2-waktu1);
    ShowMessage ('Lama Proses '+ timeToStr(selisih));
end;

procedure TForm2.bersihkanClick(Sender: TObject);
begin
        simpan.Enabled:=false;
        bersihkan.Enabled:=false;
        memo1.Lines.Clear;
        edit1.Text:=";
```

```
        edit2.Text:='';
        edit3.Text:='';
        edit4.Text:='';
        edit5.Text:='';
        edit6.Text:='';
        edit7.Text:='';
        edit8.Text:='';
        edit9.Text:='';
        edit10.Text:='';
        edit11.Text:='';
for i:=1 to 11 do
  begin
    tblist[i]:='';
  end;
jalurgm:='';
tempuh:=0;
stat:=0
end;

procedure TForm2.simpanClick(Sender: TObject);
Var
F:TextFile;
nmfile:String;
begin
If SaveDialog1.Execute Then
Begin
nmfile:=SaveDialog1.FileName+'.txt';
AssignFile(F,nmfile);
Rewrite(F);
Write(F,Memo1.Text);
CloseFile(F);
end;

end;

end.
```