PENYELESAIAN *TRAVELLING SALESMAN PROBLEM* DENGAN ALGORITMA *CHEAPEST INSERTION HEURISTICS* DAN BASIS DATA

Kusrini[1], Jazi Eko Istiyanto[2]
[1] Staf Pegajar STMIK AMIKOM Yogyakarta, Jl. Ringroad Utara Condong Catur Yogyakarta,
Telp. (0274) 884201, Faks. (0274) 884208
E-mail: kusrini@amikom.ac.id, kusrini@australia.edu
[2] Staf Pengajar Jurusan Fisika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Gadjah Mada

ABSTRAK: Ada banyak algoritma untuk memecahkan masalah *Travelling Salesman Problem* (TSP), diantaranya: *Linear Programming* (LP), Algoritma Genetik, *Nearest Neighbourhood Heuristic* (NNH) *and Cheapest Insertion Heuristic* (CIH). Makalah ini akan membahas tentang implementasi algoritma CIH untuk menyelesaikan TSP. Penulis menggunakan Borland Delphi 6 dan Interbase 6 sebagai tool dalam implementasi TSP. Algoritma CIH telah berhasil diimplementasikan. Dengan mengetahui jumlah kota yang terhubung dan jarak diantaranya, rute perjalanan dan total panjang rute untuk mengunjungi semua kota dalam jaringan dapat diketahui. Namun demikian, implementasi algoritma belum mampu menyelesaikan masalah pencarian rute jika ada 2 kota yang mimiliki bobot yang berbeda dengan melihat arahnya dan jika ada 2 buah kota yang tidak terhubung.

Kata kunci: TSP, cheapest, insertion, heuristics, basis data.

ABSTRACT: *There are plenty well-known algorithms for solving Travelling Salesman Program (TSP), such as: Linear Programming (LP), Genetic Algorithm (GA), Nearest Neighbourhood Heuristic (NNH) and Cheapest Insertion Heuristic (CIH). This paper will talk about TSP implementation by using CIH algorithm. The writer uses Borland Delphi 6 and Interbase 6 as tool for implementing TSP. CIH algorithm has been implemented successfully. By determining count of connected cities and distances between them, the traveled route and total route length to visit all cities in a cities network were obtained. However, this algorithm implementation has not yet be able to solve route searching if there are two cities have different load or there are 2 cities that is not connected.*

Keywords: *TSP, cheapest, insertion, heuristics, database*

PENDAHULUAN

Travelling Salesman Problem (TSP) merupakan masalah klasik yang mencoba mencari rute terpendek yang bisa dilalui salesman yang ingin mengunjungi beberapa kota tanpa harus mendatangi kota yang sama lebih dari satu kali. Jika jumlah kota yang harus didatangi hanya sedikit, misalnya hanya ada 5 kota, permasalahan ini dapat dipecahkan dengan sangat mudah. Bahkan tidak memerlukan komputer untuk menghitungnya. Tetapi, masalahnya jadi rumit jika ada lebih dari 20 kota yang harus didatangi. Ada begitu banyak kemungkinan yang harus dicoba dan diuji untuk menemukan jawabannya.

Beberapa contoh masalah yang dapat diselesaikan dengan menggunakan pendekatan TSP adalah [4]:

1. Pencarian rute bis sekolah untuk mengantarkan siswa
2. Pencarian rute truk pengantar parcel
3. Pengambilan tagihan telepon

Selain masalah-masalah tersebut di atas, TSP juga bisa diaplikasikan pada penjadwalan produksi [2].

Ada banyak algoritma untuk memecahkan masalah ini, diantaranya dengan menggunakan *Linier Programming (LP), Genetic Algorithm (GA), Nearest Neighbourhoud Heuristics (NNH) dan Cheapest Insertion Heuristics (CIH)* [5].

Waktu proses algoritma CIH jauh lebih cepat dibandingkan algoritma NNH [6], namun untuk memproses kota dalam jumlah besar CIH masih membutuhkan waktu yang cukup besar, karena itu Kendela, H. F. memperkenalkan algoritma Hybrid Heuristic yang menggabungkan algoritma CIH dengan algoritma NNH. Hasilnya menunjukkan waktu proses algoritma tersebut jauh lebih cepat dibadingkan dengan CIH dan NNH [3].

RUMUSAN MASALAH

1. Bagaimana membuat perangkat lunak untuk menyelesaikan kasus Travelling Salesman Problem dengan menggunakan algoritma CIH.
2. Bagaimana menerapkan algoritma CIH dengan menggabungkan dengan kemampuan query pada basis data

BATASAN MASALAH

1. Dalam makalah ini diasumsikan seluruh kota terhubung satu sama lain dengan biaya/jarak antara 2 buah kota memiliki nilai yang sama meskipun dengan arah berbeda. Misal biaya untuk menempuh jarak dari kota A ke B sama dengan biaya untuk menempuh jarak dari kota B ke A. Sementara jika ada niali biaya/jarak yang berbeda untuk arah yang berbeda dari 2 kota, belum dapat ditangani dengan aplikasi ini.
2. Bahasa pemrograman yang digunakan Borland Delphi 6 dengan dukungan DBMS Interbase 6.

TINJAUAN PUSTAKA

Penelitian mengenai TSP sudah banyak dilakukan, diantaranya:

1. Irving Vitra (2004) membandingkan metode-metode dalam algoritma genetika untuk Travelling Salesman Problem. Hasil terbaik dari pengujian pertama yang dilakukan terhadap data 19 kota adalah dengan metode kombinasi order crossover dan insertion mutation, tetapi pada pengujian selanjutnya bisa berbeda-beda [2]
2. Arna Fariza, dkk (2005) mengaplikasikan algoritma genetika multi obyektif untuk penyelesaian Travelling Salesman Problem. Dari penelitian ini, dia mendapatkan bahwa hasil perhitungan algoritma genetika dapat memberikan hasil perhitungan optimum, tetapi untuk N yang besar, solusi yang diperoleh adalah solusi yang diharapkan paling mendekati solusi sebenarnya.

METODE PENELITIAN

Dalam penelitian ini digunakan metode:

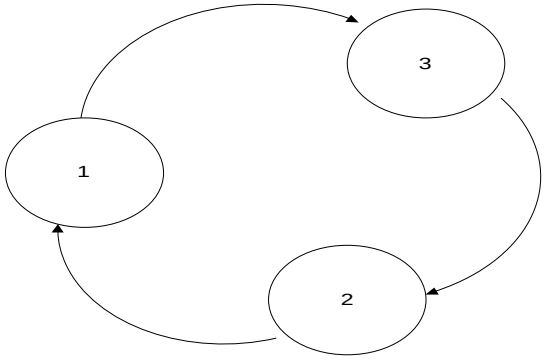
1. Perancangan algoritma
   Pada tahap awal akan dilakukan perancangan algoritma pencarian jarak terpendek dengan metode *Cheapest Insertion Heuristic* dengan menggunakan basis data.
2. Implementasi Rancangan
   Setelah algoritma tersebut terbentuk akan diimplementasikan menjadi sebuah perangkat lunak, dengan menggunakan bahasa pemrograman Delphi 6 dan basis data Interbase.
3. Uji coba
   Uji coba dilakukan untuk menguji kebenaran perhitungan, dengan membandingkan perhitungan manual dan perhitungan dengan aplikasi. Selain itu diuji pula waktu yang diperlukan untuk memproses solusi dengan jumlah kota yang berbeda-beda.

LANDASAN TEORI

Algoritma *Cheapest Insertion Heuristics* (CIH)

Berikut ini adalah tata urutan algoritma CIH [5]:

1. Penelusuran dimulai dari sebuah kota pertama yang dihubungkan dengan sebuah kota terakhir.
2. Dibuat sebuah hubungan *subtour* antara 2 kota tersebut. Yang dimaksud *subtour* adalah perjalanan dari kota pertama dan berakhir di kota pertama, misal (1,3) → (3,2) → (2,1) seperti tergambar dalam Gambar 1.



Gambar 1. *Subtour*

3. Ganti salah satu arah hubungan (*arc*) dari dua kota dengan kombinasi dua *arc*, yaitu *arc* (i,j) dengan *arc* (i,k) dan *arc* (k,j), dengan k diambil dari kota yang belum masuk *subtour* dan dengan tambahan jarak terkecil. Jarak diperoleh dari:
   $c_{ik} + c_{kj} - c_{ij}$
   c_{ik} adalah jarak dari kota i ke kota k,
   c_{kj} adalah jarak dari kota k ke kota j dan
   c_{ij} adalah jarak dari kota i ke kota j
4. Ulangi langkah 3 sampai seluruh kota masuk dalam *subtour*

Sebagai contoh diberikan 5 kota dengan jarak antar kota seperti tertera dalam Tabel 1.

Tabel 1. Jarak Antar Kota

Kota Asal	Kota Tujuan	Jarak
1	2	132
1	3	217
1	4	164
1	5	58
2	3	290
2	4	201
2	5	79
3	4	113
3	5	303
4	5	196

Untuk mencari jarak terpendek melalui ke 5 kota tersebut sebagaimana terdapat dalam Tabel 1, ambil langkah-langkah sebagai berikut:

1. Ambil perjalanan dari kota 1 ke 5
2. Buat *subtour* → (1,5) → (5,1)
3. Buat tabel yang menyimpan kota yang bisa disisipkan dalam *subtour* beserta tambahan jaraknya, seperti ditampilkan dalam Tabel 2.

Tabel 2. *Arc* Penambah *Subtour* ke 1

Arc yang akan diganti	*Arc* yang ditambahkan ke *subtour*	Tambahan Jarak
(1,5)	(1,2) – (2,5)	$c_{12} + c_{25} - c_{15} = 153$
(1,5)	(1,3) – (3,5)	$c_{13} + c_{35} - c_{15} = 462$
(1,5)	(1,4) – (4,5)	$c_{14} + c_{45} - c_{15} = 302$
(5,1)	(5,2) – (2,1)	$c_{52} + c_{21} - c_{51} = 153$
(5,1)	(5,3) – (3,1)	$c_{53} + c_{31} - c_{51} = 462$
(5,1)	(5,4) – (4,1)	$c_{54} + c_{41} - c_{51} = 302$

Dari tabel 2 diberoleh tambahan jarak terkecil apabila:
Arc (1,5) diganti dengan *arc* (1,2) dan *arc* (2,5) atau
arc (5,1) diganti dengan *arc* (5,2) dan *arc* (2,1) dari kemungkinan tersebut, bisa dipilih salah satu. Misal dipilih kemungkinan pertama maka *subtour* yang baru menjadi:
→ (1,2) → (2,5) → (5,1)

4. Selanjutnya dibuat tabel yang menyimpan kota yang bisa disisipkan dalam *subtour* beserta tambahan jaraknya, seperti ditampilkan dalam tabel 3.

Tabel 3. *Arc* Penambah *Subtour* ke 2

Arc yang akan diganti	*Arc* yang ditambahkan ke *subtour*	Tambahan Jarak
(1,2)	(1,3) – (3,2)	$c_{13} + c_{32} - c_{12} = 375$
(1,2)	(1,4) – (4,2)	$c_{14} + c_{42} - c_{12} = 233$
(2,5)	(2,3) – (3,5)	$c_{23} + c_{35} - c_{25} = 514$
(2,5)	(2,4) – (4,5)	$c_{52} + c_{21} - c_{51} = 318$
(5,1)	(5,3) – (3,1)	$c_{53} + c_{31} - c_{51} = 462$
(5,1)	(5,4) – (4,1)	$c_{54} + c_{41} - c_{51} = 302$

Dari tabel 3 diberoleh tambahan jarak terkecil adalah 233 dengan menggantikan *arc*(1,2) dengan *arc*(1,4) dan *arc*(4,2), sehingga *subtour* baru yang dihasilkan adalah:
→ (1,4) → (4,2) → (2,5) → (5,1)

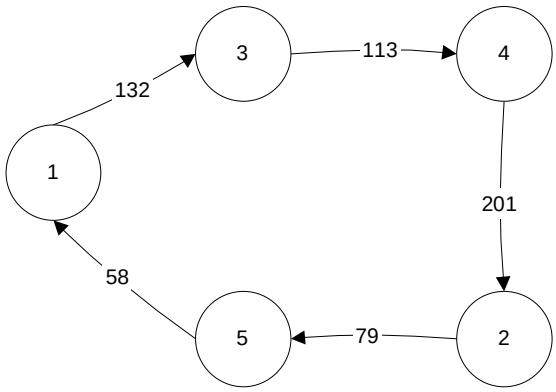
5. Karena masih ada kota yang belum masuk, perlu dibuat tabel yang menyimpan kota yang bisa disisipkan dalam *subtour* beserta tambahan jaraknya, seperti ditampilkan dalam Tabel 4.

Tabel 4. *Arc* Penambah *Subtour* ke 3

Arc yang akan diganti	*Arc* yang ditambahkan ke *subtour*	Tambahan Jarak
(1,4)	(1,3) – (3,4)	$c_{13} + c_{34} - c_{14} = 166$
(4,2)	(4,3) – (3,2)	$c_{43} + c_{32} - c_{42} = 202$
(2,5)	(2,3) – (3,5)	$c_{23} + c_{35} - c_{25} = 514$
(5,1)	(5,3) – (3,1)	$c_{53} + c_{31} - c_{51} = 462$

Dari tabel 4 diberoleh tambahan jarak terkecil adalah 166 dengan menggantikan *arc* (1,4) dengan *arc* (1,3) dan *arc* (3,4), sehingga *subtour* baru yang dihasilkan adalah:
→ (1,3) → (3,4) → (4,2) → (2,5) → (5,1)

Dari langkah-langkah tersebut diatas dapat diperoleh lintasan terpendek untuk mengunjungi 5 kota adalah → (1,3) → (3,4) → (4,2) → (2,5) → (5,1) seperti terlihat pada Gambar 2.



Gambar 2. Lintasan Terpendek antar 5 Kota

Dengan lintasan tersebut diperoleh jarak tempuhnya adalah:
$c_{13} + c_{34} + c_{42} + c_{25} + c_{51}$ = 132 + 113 + 201 + 79 + 58 = 668

PERANCANGAN

Input dari program TSP dengan CIH adalah:
1. Jumlah kota
2. Jarak antar kota

Sedangkan output program adalah sebagai berikut:
1. Proses pemilihan lintasan
2. Lintasan terpendek
3. Jarak lintasan

Untuk mengimplementasikan algoritma CIH, penulis menggunakan bantuan basis data. Adapun rancangan basis data yang digunakan adalah seperti tertera pada Gambar 3.

Tabel Jarak		
Nama	**Tipe**	**PK**
Asal	Int	Y
Tujuan	Int	Y
Jarak	Int	

Tabel Proses		
Nama	**Tipe**	**PK**
Asal	Int	Y
Sisip	Int	Y
Tujuan	Int	Y
Jarak	Int	

Tabel Hasil		
Nama	**Tipe**	**PK**
Asal	Int	Y
Tujuan	Int	Y
Nomor	Int	

Gambar 3. Rancangan Basis Data CIH

Tabel jarak digunakan untuk menyimpan data kota-kota yang akan dikunjungi beserta jarak antar kota.

Tabel proses digunakan sebagai fasilitas temporary dalam penyisipan *subtour*. Penggunaan tabel ini akan mempermudah dalam pengambilan informasi jarak minimal dari beberapa alternatif yang ada. Dalam pencarian jarak minimal digunakan query dengan fungsi agregasi *min*. Sedangkan tabel hasil digunakan untuk menyimpan data hasil penemuan kota

Adapun algoritma pencarian lintasan setelah ditentukan jumlah kota yang dikunjungi dan jarak antar kota, tampak dalam flowchart seperti Gambar 4.

Untuk menggunakan program ini, pengguna harus melakukan langkah sebagai berikut:

1. Menentukan jumlah kota. Program akan secara otomatis memberikan tempat pengisian jarak antar kota sesuai jumlah kota yang dimasukkan
2. Isikan jarak antar kota
3. Tekan tombol Proses

Dengan menggunakan kasus 5 kota dengan jarak seperti tertera pada tabel 1 diperoleh lintasan → (1,3) → (3,4) → (4,2) → (2,5) → (5,1). Lintasan tersebut sama dengan yang tertera pada hasil hitungan manual pada Gambar 2. Jarak lintasan adalah 668.

Dengan menggunakan aplikasi ini jumlah kota yang akan dilalui bisa sangat banyak. Untuk merubah jumlah kota, tinggal memasukkan di tempat yang sudah disediakan.

Dengan menambahkan fungsi random untuk mengisi jarak antar kota, aplikasi ini telah diuji dengan beberapa jumlah kota, menghasilkan waktu proses sebagai seperti tampak pada Tabel 5

Tabel 5. Waktu Proses untuk n Jumlah Kota

Jumlah Kota	Waktu proses (detik)
5	1
10	3
15	7
20	17
25	33
30	56
35	87

Untuk menghitung estimasi waktu yang diperlukan jika untuk menghitung jumlah kota dengan aplikasi ini, akan digunakan trend linier dengan rumus sebagai berikut [1]:

$$Y = a + bX$$

dengan

$$a = \frac{\sum Y}{n} \text{ dan } b = \frac{\sum XY}{\sum X^2}$$

Berikut ini adalah perhitungan trend untuk mencari waktu dengan diketahui jumlah kotanya:

No	Y	X	XY	XX
1	1	-3	-3	9
2	3	-2	-6	4
3	7	-1	-7	1
4	17	0	0	0
5	33	1	33	1
6	56	2	112	4
7	87	3	261	9
	204	**0**	**390**	**28**

$N = 7$, $\sum Y = 204$, $\sum X = 0$, $\sum XY = 390$, $\sum XX = 28$

$$a = \frac{204}{7} = 29$$

$$b = \frac{390}{28} = 14$$

sehingga $Y = 29 + 14X$

dengan

Y = waktu proses

X = (Jumlah kota-20)/5

Atau jika X diisi dengan jumlah kota, maka persamaannya akan menjadi:

$$Y = \frac{14}{5} X - 27$$

KESIMPULAN

Berdasarkan proses pembuatan, dan hasil ujicoba dapat disimpulkan bahwa:

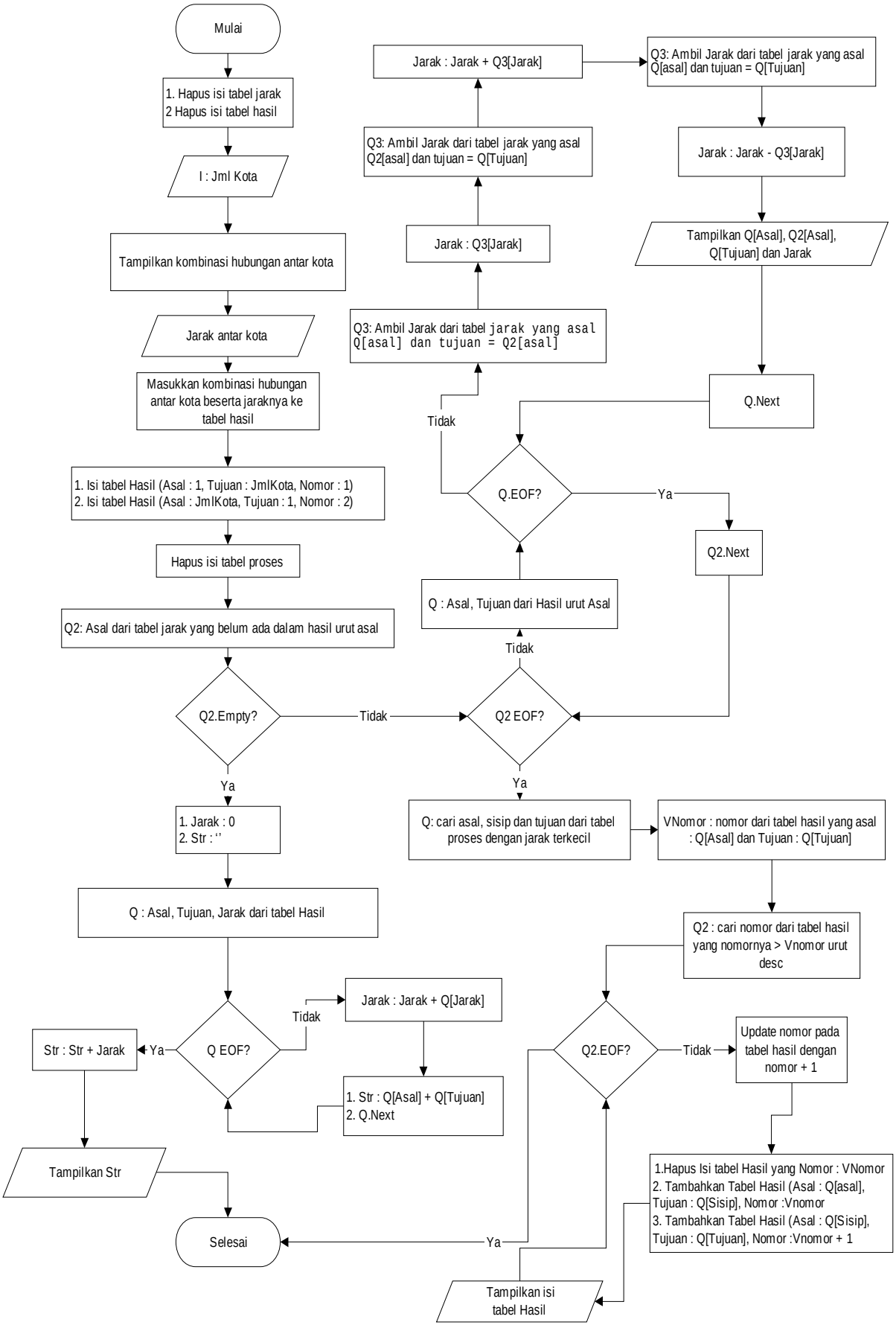
1. Basis data dapat digunakan sebagai alat bantu dalam mengimplementasikan algoritma cheapest insertion heuristic dengan baik.
2. Peran basis data dalam implementasi ini adalah untuk penyimpanan data proses sehingga pengambilan informasi jarak minimal dari beberapa alternatif yang ada dapat dilakukan dengan mudah yaitu dengan menggunakan query.
3. Banyaknya jumlah kota sangat mempengaruhi waktu proses penyelesaian. Fungsi waktu proses terhadap jumlah kota adalah:

   $$Y = \frac{14}{5} X - 27$$

   dengan

   Y = waktu proses (detik)

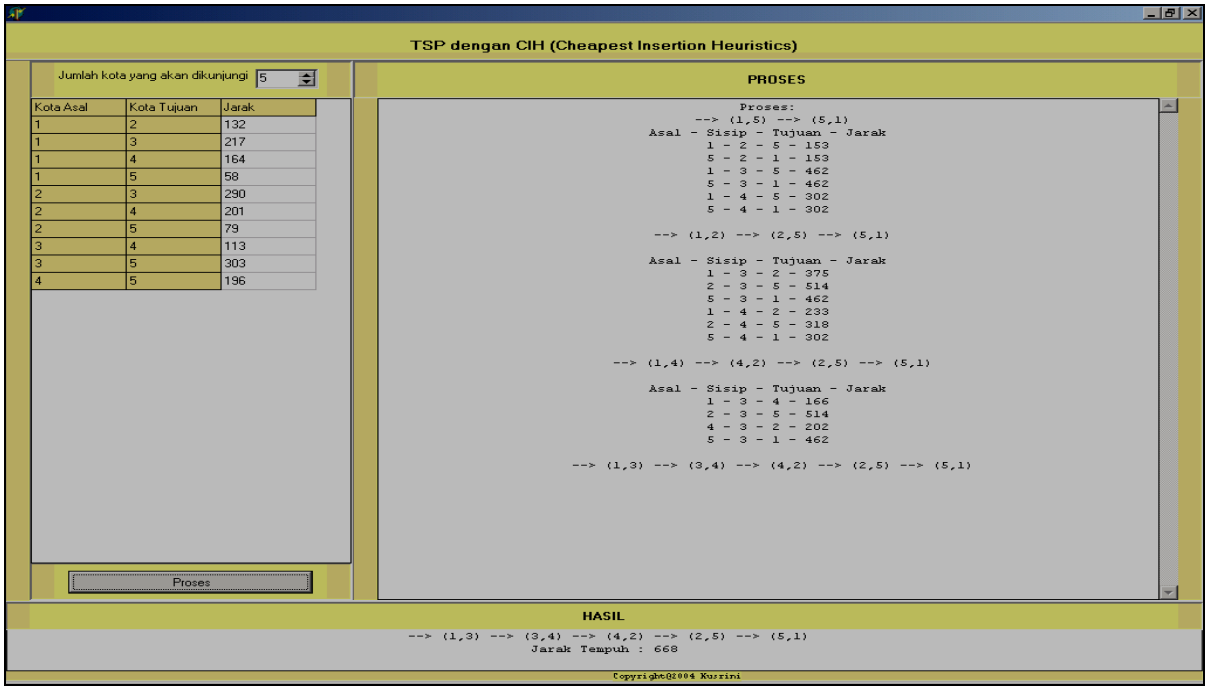
   X = jumlah kota



Gambar 4. Flowchart Program

HASIL DAN PEMBAHASAN

Pada Gambar 5 berikut ini adalah tampilan hasil program TSP dengan CIH.



Gambar 5. Tampilan hasil program

SARAN

Aplikasi ini dapat dikembangkan untuk kasus hubungan antar kota yang tidak selalu bolak-balik.

DAFTAR PUSTAKA

1. Erlina, *Peramalan anggaran perjualan*, USU digital Lybrary, 2002.
2. Farida A., dkk, *Aplikasi Algoritma Genetika Multi Obyektif pada Travelling Salesman Problem*, Prosiding Seminar Nasional "Soft Computing, Intelligent Systems and Information Technology" 2005..
3. Kendela, H.F., Al-Ahmar, M. A., Horbaty, E.M., *A Hibrid Heuristic Algorithm for the Travelling Salesman Problem*, 2006.
4. Osborn, D., *lnteger Programming Application*, courses.washington.edu/ie312or/Lecture-IP-TSP.pdf, tanggal akses 24 Februari 2007
5. Vitra, I., *Perbandingan metode-metode dalam algoritma genetika untuk Travelling Salesman Problem.* Proceedings Seminar Nasional Aplikasi Teknologi Informasi 2004.
6. Winston, WL, *Operation Research,* Duxburry, 1997,