PERBANDINGAN ALGORITMA *GREEDY* , ALGORITMA *CHEAPEST INSERTION HEURISTICS* DAN *DYNAMIC PROGRAMMING* DALAM PENYELESAIAN *TRAVELLING SALESMAN PROBLEM*

Gea Aristi
Program Studi Manajemen Informatika
AMIK BSI Tasikmalaya
Jl.Tanuwijaya No.4 Empangsari Tawang Tasikmalaya 46113 Indonesia
gea.gas@bsi.ac.id

ABSTRACT

Travelling Salesman Problem is one of problems to find shortest route from travelling a salesman from first city and then to destination cities and finally back to first city, but one city just only once visited. There are some algorithms to solving travelling salesman problem,such as Greedy Algorithm, Artificial Bee Colony Algorithm, Cheapest Insertion Heuristics Algorithm, Genetic Algorithm and many more. In this paper, only greedy algorithm, cheapest insertion heuristics algorithm and dynamic programming are discussed. After compared using an example case with 5 cities and solved by third of algorithm, shortest route is same but the way to solving is different. They have advantages and disadvantages of each, and has the characteristics of each. Greedy algorithm is more suitable when used for a number of cities that are not too much because the process is more simple, but cheapest insertion heuristics algorithm is more suitable to case with more city throught the process more complicated than greedy algorithm. Counting in Dynamic programming must be right because it will be influence for the next counting result.

Keywords*: Cheapest Insertion Heuristic*s*, Greedy, Dynamic Programming, Travelling Salesman Problem.*

I. PENDAHULUAN

Ketika kita berpergian dari suatu tempat ke tempat lain maka kita mempertimbangkan waktu seefisien mungkin dan biayanya sehingga kita mencari jalur terpendek yang akan ditempuh. Permasalahan tersebut dikenal dengan nama *Travelling Salesman Problem*. *Travelling Salesman Problem* (TSP) adalah perjalanan seorang *salesman* menuju ke semua tempat yang akan dituju dan akhirnya kembali ke tempat awal dengan menggunakan jalur terpendek, akan tetapi semua tempat hanya boleh dilalui satu kali. Pada permasalahan *Travelling Salesman Problem* ini dicari sebuah solusi yaitu menghitung jalur terpendek atau rute perjalanan minimum dari semua lintasan.

Terdapat beberapa algoritma yang digunakan untuk menemukan jalur terpendek dari *travelling salesman problem* tersebut. Algoritma yang digunakan diantaranya adalah Algoritma *Greedy*, Algoritma *Artificial Bee Colony*, Algoritma *Cheapest Insertion Heuristics*, Algoritma Genetika dan masih banyak yang lainnya. Masing-masing algoritma memiliki kelebihan dan kekurangannya. Suatu algoritma memiliki hasil yang berbeda-beda, karena belum tentu suatu algoritma yang memiliki optimasi yang tinggi untuk suatu kasus memiliki optimasi yang tinggi pula untuk kasus yang lain.

Dengan demikian maka pada pembahasan kali ini akan dibandingkan hasil dari perhitungan dengan menggunakan algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynamic programming* yang nantinya akan diketahui algoritma mana yang lebih baik untuk menyelesaikan kasus *Travelling Salesman Problem* yang ada.

II. KAJIAN LITERATUR

A. *Travelling Salesman Problem*

Travelling Salesman Problem adalah salah satu contoh persoalan optimasi yang mengaplikasi teori graf. Bentuk umum *Travelling Salesman Problem* pertama kali dipelajari oleh para matematikawan pada tahun 1930 yang diawali oleh Karl Menger di Vienna dan Harvard. Setelah itu permasalahan *Travelling Salesman Problem* dipublikasikan oleh Hassler Whitney dan Merrill Flood di Pricenton.

Menurut Dian (2013:2), *Travelling Salesman Problem* dikatakan ada dua jenis, yaitu:

1. *Travelling Salesman Problem* asimetris
   Pada *Travelling Salesman Problem* jenis ini, biaya dari kota 1 ke kota 2 tidak sama dengan biaya dari kota 2 ke kota 1.

Dengan n kota, besarnya ruang pencarian adalah $\frac{n!}{n}$ = (n - 1)! jalur yang mungkin.

2. *Travelling Salesman Problem* simetris
   Sedangkan pada *Travelling Salesman Problem* jenis simetris, biaya dari kota 1 ke kota 2 adalah sama dengan biaya dari kota 2 ke kota 1. Apabila dengan n kota, jumlah jalur yang mungkin adalah:
   $\frac{n!}{2n} = \frac{(n-1)!}{2}$ jalur yang mungkin.

B. Algoritma *Greedy*

Algoritma *greedy* adalah salah satu algoritma yang biasa digunakan untuk memecahkan masalah optimasi. Menurut Dian (2013:2) algoritma *Greedy* merupakan algoritma yang dapat menentukan sebuah jalur terpendek antara *node-node* yang akan digunakan dengan mengambil secara terus menerus dan menambahkannya ke dalam jalur yang akan dilewati. Mengacu pada konsep *greedy* yang menganggap bahwa pada setiap langkah akan dipilih tempat atau kota yang belum pernah dikunjungi, dimana tempat atau kota tersebut memiliki jarak terdekat dari tempat atau kota sebelumnya. Algoritma ini tidak mempertimbangkan nilai heuristic, yang dalam hal ini bisa berupa jarak langsung antar dua tempat.

Penyelesaian *Travelling Salesman Problem* dengan algoritma *greedy* adalah sebagai berikut: misalnya tujuan dipresentasikan dengan angka 1 hingga n, tujuan 1 merupakan awal atau titik awalan dimana seorang *salesman* pergi ke tujuan. Asumsikan fungsi C(i,j) adalah biaya tujuan dari kota i ke j, ada kemungkinan tidak sama dengan biaya C(j,i). Kemungkinan solusi yang pasti adalah: (n-1)!.

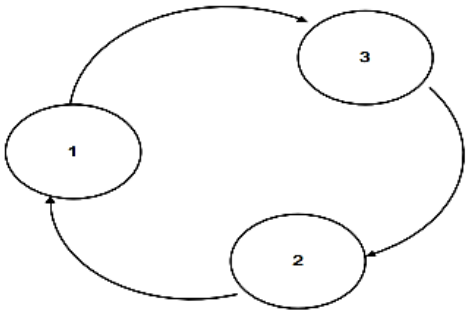
Algoritma *greedy* memiliki tahapan-tahapan yang perlu di perhatikan. Diantaranya adalah sebagai berikut :

1. Mulai dari sembarang tujuan, jika tempat asal telah ditetapkan maka mulailah dari tempat tujuan tersebut.
2. Lakukan evaluasi terhadap biaya menuju tujuan.
3. Pilih tujuan dengan biaya terkecil dan diulang pada langkah ke dua hingga tujuan telah terlewati semuanya.
4. Hitung jumlah tujuan yang telah dilewati.

C. Algoritma *Cheapest Insertion Heuristics*

Algoritma *Cheapest Insertion Heuristics* adalah algoritma yang membangun suatu *tour* dari sikel-sikel kecil dengan bobot minimal dan secara berturut-turut ditambah dengan titik baru sampai semua titik berhasil dilalui. Berikut ini adalah tata cara urutan algoritma CIH berdasarkan *paper* Winston, Wayne L. dan Goldberg, Jeffrey B pada tahun 2004 yang berjudul *Operations Research Application And Algorithms 4th Edition*

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. Seperti (1,3) ➔ (3,2) ➔ (2,1) pada gambar 1.



Sumber: Dian (2013:3)

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
   - C_{ij} adalah jarak dari kota i ke kota j
4. Ulangi langkah 3 sampai seluruh kota masuk dalam *subtour*.

D. *Dynamic Programming*

Menurut Horowitz dalam Dian (2008) *Dynamic Programming* adalah sebuah metode desain algoritma yang dapat digunakan ketika solusi untuk sebuah masalah yang mungkin terlihat sebagai hasil dari sebuah urutan keputusan[2]. *Dynamic Programming* dapat dikatakan pula sebagai metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan langkah (*step*) atau tahapan (*stage*) sedemikian rupa sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan. Pada penyelesaian persoalan dengan metode ini:

1) Terdapat sejumlah berhingga pilihan yang mungkin.
2) Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya .
3) Menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Metode/teknik pemrograman dinamik pada TSP dapat dijelaskan sebagai berikut. Misalkan kita harus menjalankan rute sebuah mobil pos untuk mengambil surat-surat dari kotak surat yang berada pada n tempat yang berbeda. Suatu graph dengan n+I vertex dapat digunakan untuk menggambarkan situasi itu. Satu vertex menunjukkan kantor pos dari mana mobil pos berangkat, dan harus kembali ketempat itu juga. Edge<ij> diberi bagi suatu nilai yang sama dengan jarak dari tempat i ke tempat j. Rute yang ditempuh mobil pos itu adalah sebuah rute dan kita ingin mendapatkan suatu rute yang panjangnya minimum.

Perjalanan path sederhana dimulai dan diakhiri di vertex 1. Setiap perjalanan terdiri dari edge <I,k> untuk k E V - {I} dan path dari vertex k sampai vertex I. Path dari vertex k sampai vertex I melewati setiap vertex dalam V - { I.k } persis satu kali. Untuk mudahnya jika perjalanan adalah optimal, maka path dari k sampai 1 harus menurun dari k sarnpai 1 path melewati setiap vertex dalam V - { 1,k }. Karena itu dasarnya adalah optimasi. Andaikan g (i,S) lenght dari path di mulai di vertex i, melewati setiap vertex dalam S dan berhenti di vertex 1. Misalkan g(1, V - { 1 }) adalah panjang optimal dari pedagang keliling. Ikuti prinsip optimasi itu :

$$g(1, V - \{1\}) = \min \{ C_{1k} + g(k, V - \{1,k\}) \}$$
$$2 <= k < n \qquad (1)$$

Secara umum (1) kita peroleh :

$$g(i,S) = \min \{ C_{ij} + g(j, S - \{j\}) \} \qquad (2)$$

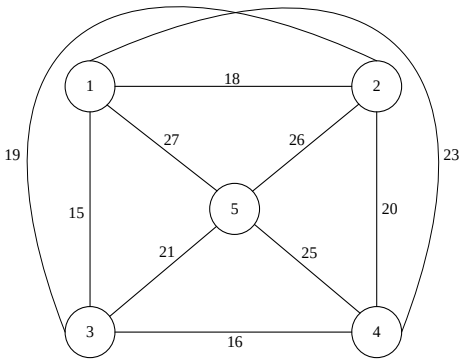
III. METODE PENELITIAN

Penelitian ini dilakukan dengan langkah-langkah sebagai berikut:

1. Menguraikan konsep algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynamic programming* untuk mencari lintasan terpendek.
2. Mengimplementasikan algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynamic programming* dalam suatu kasus.
3. Melakukan perbandingan dari algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynamic programming* sesuai dari hasil penyelesaian kasus yang ada.
4. Membuat kesimpulan algoritma mana yang cocok untuk kasus tersebut.

IV. PEMBAHASAN

Travelling Salesman Problem bisa diselesaikan dengan beberapa algoritma, diantaranya adalah algoritma *greedy*, algoritma *artificial bee colony*, algoritma *cheapest insertion heuristics*, algoritma genetika dan masih banyak yang lainnya. Namun pada pembahasan kali ini algoritma yang digunakan adalah algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynamic programming* Ketiga algoritma tersebut diimplementasikan ke dalam sebuah contoh untuk mengetahui perbandingan dari ketiga algoritma tersebut. Untuk contoh kasusnya dapat dilihat pada gambar 2. Seperti terlihat pada gambar 2 terdapat 5 buah kota yang masing-masing tempat mempunyai jarak yang berbeda-beda. Untuk kota awal sudah ditentukan yaitu tempat no 1.



sumber : Hasil penelitian (2014)

Gambar 2. Skema jalur

Jarak antara kota yang satu dengan kota yang lainnya dapat dilihat pada tabel 1 di bawah ini:

Tabel 1. Jarak Antar Kota

Kota Asal	Kota Tujuan	Jarak
1	2	18
1	3	15
1	4	23
1	5	27
2	3	19
2	4	20
2	5	26
3	4	16
3	5	21
4	5	25

Sumber : hasil penelitian (2014)

Berikut cara penyelesaian kasus tersebut dengan menggunakan algoritma *greedy*,

algoritma *cheapest insertion heuristic*s dan *dynamic programming.*

A. Algoritma *Greedy*

Cara penyelesaian *travelling salesman problem* dengan algoritma *greedy* adalah:

1. Mulai dari sembarang tujuan, jika tempat asal telah ditetapkan maka mulailah dari tempat tujuan tersebut. Tempat asalnya adalah no.1
2. Kemungkinan solusi yang muncul adalah (n-1)! = (5-1)! = 24, maka kemungkinan jalurnya adalah:

Tabel 2. Kemungkinan Jalur

Langkah ke-	Jalur	Total Jarak
1	1-2-3-4-5-1	18+19+16+25+27 = 105
2	1-2-3-5-4-1	18+19+21+25+23 = 106
3	1-2-4-5-3-1	18+20+25+21+15 = 99
4	1-2-4-3-5-1	18+20+16+21+27 = 102
5	1-2-5-4-3-1	18+26+25+16+15 = 100
6	1-2-5-3-4-1	18+26+21+16+23 = 104
7	1-3-4-5-2-1	15+16+25+26+18 = 100
8	1-3-4-2-5-1	15+16+20+26+27 = 104
9	1-3-5-2-4-1	15+21+26+20+23 = 105
10	1-3-5-4-2-1	15+21+25+20+18 = 99
11	1-3-2-4-5-1	15+19+20+25+27 = 106
12	1-3-2-5-4-1	15+19+26+25+23 = 108
13	1-4-5-2-3-1	23+25+26+19+15 = 108
14	1-4-5-3-2-1	23+25+21+19+18 = 106
15	1-4-2-3-5-1	23+20+19+21+27 = 110
16	1-4-2-5-3-1	23+20+26+21+15 = 105
17	1-4-3-5-2-1	23+16+21+26+18 = 104
18	1-4-3-2-5-1	23+16+19+26+27 = 111
19	1-5-2-3-4-1	27+26+19+16+23 = 111
20	1-5-2-4-3-1	27+26+20+16+15 = 104
21	1-5-3-4-2-1	27+21+16+20+18 = 102
22	1-5-3-2-4-1	27+21+19+20+23 = 110
23	1-5-4-2-3-1	27+25+20+19+15 = 106
24	1-5-4-3-2-1	27+25+16+19+18 = 105

Sumber : sumber penelitian (2014)

Berdasarkan hasil perhitungan dengan algoritma *greedy* diatas maka jalur yang terpendek ada dua buah kemungkinan jalur yaitu: 1-2-4-5-3-1 atau 1-3-5-4-2-1 dengan total jarak 99km

B. Algoritma *Cheapest Insertion Heuristics*

Berdasarkan tahapan-tahapan pada algoritma *cheapest insertion heuristic*s maka penyelesaian kasus pada gambar 2 tersebut adalah:

1. Penelusuran dimulai dari sebuah kota pertama yang dihubungkan dengan sebuah kota terakhir yaitu dari kota 1 ke kota 5
2. Dibuat sebuah hubungan *subtour* antara 2 kota tersebut : (1,5) ➔ (5,1)
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
   - ✓ C_{ij} adalah jarak dari kota i ke kota j

Maka dibuatlah tabel 3 yang menyimpan kota yang bisa disisipkan dalam *subtour* beserta tambahannya.

Tabel 3. *Arc* Penambah *Subtour* ke1

Arc yang akan diganti	*Arc* yang ditambahkan ke *subtour*	Tambahan Jarak
(1,5)	(1,2)-(2,5)	$C_{12}+C_{25}-C_{15}$=18+26-27=17
(1,5)	(1,3)-(3,5)	$C_{13}+C_{35}-C_{15}$=15+21-27=9
(1,5)	(1,4)-(4,5)	$C_{14}+C_{45}-C_{15}$=23+25-27=21
(5,1)	(5,2)-(2,1)	$C_{52}+C_{21}-C_{51}$=26+18-27=17
(5,1)	(5,3)-(3,1)	$C_{53}+C_{31}-C_{51}$=21+15-27=9
(5,1)	(5,4)-(4,1)	$C_{54}+C_{41}-C_{51}$=25+23-27=21

Sumber : hasil penelitian (2014)

Berdasarkan tabel 3 diatas maka dapat diperoleh jarak yang terkecil adalah 9, apabila *arc* (1,5) diganti dengan *arc* (1,3) dan *arc* (3,5) atau *arc* (5,1) diganti dengan *arc* (5,3) dan *arc* (3,1). Karena terdapat dua jarak yang terkecil maka dari kemungkinan tersebut bisa dipilih salah satu, misalkan kemungkinan pertama maka *subtour* (1,5) ➔ (5,1) diganti menjadi (1,3) ➔ (3,5) ➔ (5,1).

Karena masih ada kota yang belum masuk, maka dibuat kembali seperti hasil dari tabel 3.

Tabel 4. *Arc* Penambah *Subtour* ke2

Arc yang akan diganti	*Arc* yang ditambahkan ke *subtour*	Tambahan Jarak
(1,3)	(1,2)-(2,3)	$C_{12}+C_{23}-C_{13}$=18+19-15=22
(1,3)	(1,4)-(4,3)	$C_{14}+C_{43}-C_{13}$=23+16-15=24
(3,5)	(3,2)-(2,5)	$C_{32}+C_{25}-C_{35}$=19+26-21=24
(3,5)	(3,4)-(4,5)	$C_{34}+C_{45}-C_{35}$=16+25-21=20
(5,1)	(5,2)-(2,1)	$C_{52}+C_{21}-C_{51}$=26+18-27=17
(5,1)	(5,4)-(4,1)	$C_{54}+C_{41}-C_{51}$=25+23-27=21

Sumber : hasil penelitian (2014)

Dari tabel 4 diperoleh tambahan jarak terkecil adalah 17 dengan mengganti *arc* (5,1) dengan *arc* (5,2) dan *arc* (2,1). Sehingga *subtour* yang baru dihasilkan (1,3) ➔ (3,5) ➔ (5,2) ➔ (2,1)

Langkah tersebut diulang karena masih ada kota yang belum masuk seperti pada tabel 5

Tabel 5. *Arc* Penambah *Subtour* ke3

Arc yang akan diganti	*Arc* yang ditambahkan ke *subtour*	Tambahan Jarak
(1,3)	(1,4)-(4,3)	$C_{14}+C_{43}-C_{13}$=23+16-15=24
(3,5)	(3,4)-(4,5)	$C_{34}+C_{45}-C_{35}$=16+25-21=20
(5,2)	(5,4)-(4,2)	$C_{54}+C_{42}-C_{52}$=25+20-26=19
(2,1)	(2,4)-(4,1)	$C_{24}+C_{41}-C_{21}$=20+23-18=25

Sumber: hasil penelitian (2014)

Setelah semua kota sudah masuk maka penambahan *subtour* selesai dan diperoleh jarak terkecil yaitu 19 dengan mengganti *arc*

(5,2) dengan *arc*(5,4) dan *arc*(4,2). Dengan demikian *subtour* yang baru adalah (1,3) ➔ (3,5) ➔ (5,4) ➔ (4,2) ➔ (2,1) sedangkan jaraknya adalah 15+21+25+20+18 = 99km

C. *Dynamic Programming*

Berdasarkan cara *dynamicProgramming* maka *travelling salesman problem* dapat diselesaikan dengan cara sebagai berikut:

Tabel 6. Panjang rute

	1	2	3	4	5
1	**-**	**18**	**15**	**23**	**27**
2	**18**	**-**	**19**	**20**	**26**
3	**15**	**19**	**-**	**16**	**21**
4	**23**	**20**	**16**	**-**	**25**
5	**27**	**26**	**21**	**25**	**-**

Sumber : hasil penelitian (2014)

Notasi-notasi yang digunakan adalah:
N = Banyaknya tahap
n = Indeks untuk tahap sekarang (n=1,2,...N)
S_n = Keadaan sekarang untuk tahap n
S_{nk} = Keadaan k yang mungkin ditempuh pada tahap n (k Є bilangan bulat positif)
C_{Sn} = Biaya pada tahap n
S_0 = Kota asal
X_n = Peubah keputusan untuk tahap n
X^*_n = Nilai Optimal X_n (diketahui S_n)
$F_n(S_n, X_n)$ = Kontribusi tahap n, n+1, ..., N

Keterangan:
S_{11} = kota ke-1 yang mungkin dilalui pada tahap 1
S_{12} = kota ke-2 yang mungkin dilalui pada tahap 1
.
.
.
S_{nk} = kota ke-k yang mungkin untuk dilaluui pada tahap n dengan k Є bilangan bulat positif; n= 1,2,3,...

Dengan demikian kasus *Travelling Salesman Program* tersebut diimplementasikan dalam notasi-notasi dinamik dengan N=5 ; n = 1,2,3,4,5. Karena dianalisa secara rekursif mundur, maka dimulai dari tahap akhir atau tahap 5.

Tabel 7 Tahapan ke-5

S	$f^*_5(s)$	x^*_5
S_{41}	18	S5
S_{42}	15	S5
S_{43}	23	S5
S_{44}	27	S5

Sumber : hasil penelitian (2014)

Untuk tahapan berikutnya nilainya mengacu dari hasil tahapan ke-5. Maka untuk tahapan ke-4 jika dihitung dengan menggunakan cara *dynamic programming* dapat dilihat pada tabel 8 di bawah ini:

Tabel 8. Tahapan ke-4

X_4 / S	$f_4(S, X_4) = C_{sx4} + f^*_5(x_4)$				$f^*_4(s)$	x^*_4
	S_{41}	S_{42}	S_{43}	S_{44}		
S_{31}	-	15+19	23+20	27+26		
S_{32}	18+19	-	23+16	27+21		
S_{33}	18+20	15+16	-	27+25		
S_{34}	18+26	15+21	23+25	-		

Sumber : hasil penelitian (2014)

Atau jika nilai tersebut dihitung maka dapat dilihat pada tabel 9 di bawah ini,

Tabel 9. Hasil Perhitungan Tahapan ke-4

X_4 / S	$f_4(S, X_4) = C_{sx4} + f^*_5(x_4)$				$f^*_4(s)$	x^*_4
	S_{41}	S_{42}	S_{43}	S_{44}		
S_{31}	-	34	43	53	34	S_{42}
S_{32}	37	-	39	48	37	S_{41}
S_{33}	38	31	-	52	31	S_{42}
S_{34}	44	36	48	-	36	S_{42}

Sesuai dengan hasil dari tahapan ke-4 maka diperoleh nilai f^*_4(s) paling minimum adalah yang melalui S_{42} dengan nilai f^*4(s) = 31. Dengan demikian pada tahapan ini yang diambil adalah rute S_{42}. Selanjutnya untuk tahapan ke-3 dapat dilihat pada tabel 10 di bawah ini.

Tabel 10. Tahapan ke-3

X_3 / S	$f_3(S, X_3) = C_{sx3} + f^*_4(x_3)$				$f^*_3(s)$	x^*_3
	S_{31}	S_{32}	S_{33}	S_{34}		
S_{21}	-	37+19	31+20	36+26		
S_{22}	34+19	-	31+16	36+21		
S_{23}	34+20	37+16	-	36+25		
S_{24}	34+26	37+21	31+25	-		

Sumber : hasil penelitian (2014)
Sedangkan untuk hasil yang telah dihitung dapat dapat dilihat pada tabel 11

Tabel 11. Hasil Perhitungan Tahapan ke-3

X_3 / S	$f_3(S, X_3) = C_{sx3} + f^*_4(x_3)$				$f^*_3(s)$	x^*_3
	S_{31}	S_{32}	S_{33}	S_{34}		
S_{21}	-	56	51	62	51	S_{33}
S_{22}	53	-	47	57	47	S_{33}
S_{23}	54	53	-	61	53	S_{32}
S_{24}	60	58	56	-	56	S_{33}

Sumber : hasil penelitian (2014)
Pada tahapan ke-3 ini nilai dari f^*_3(s) yang paling minimum adalah yang melalui S_{33} dengan nilai f^*_3(s)= 47. Maka rute yang diambil adalah S_{33}. Untuk tahapan selanjutnya dapat dilihat pada tabel 12.

Tabel 12. Tahapan ke-2

X_2 / S	$f_2(S, X_2) = C_{sx2} + f^*_3(x_2)$				$f^*_2(s)$	x^*_2
	S_{21}	S_{22}	S_{23}	S_{24}		
S_{11}	-	47+19	53+20	56+26		

S_{12}	51+19	-	53+16	56+21		
S_{13}	51+20	47+16	-	56+25		
S_{14}	51+26	47+21	53+25	-		

Sumber : hasil penelitian (2014)

Tabel 13. Hasil Perhitungan Tahapan ke-2

X_2 / S	$f_2(S, X_2) = C_{sx2} + f^*_3(x_2)$				$f^*_2(s)$	x^*_2
	S_{21}	S_{22}	S_{23}	S_{24}		
S_{11}	-	66	73	82	66	S_{22}
S_{12}	70	-	69	77	69	S_{23}
S_{13}	71	63	-	81	63	S_{22}
S_{14}	77	68	78	-	68	S_{22}

Sumber : hasil penelitian (2014)

Hasil dari tahapan ke-2 maka diperoleh nilai $f^*_2(s)$ paling minimum adalah yang melalui $S_{22.}$ dengan nilai $f^*2(s) = 63$. Dengan demikian pada tahapan ini yang diambil adalah rute S_{22}. Untuk tahapan ke-1 dapat dilihat pada tabel 14.

Tabel 14. Tahapan ke-1

X_1 / S	$f_1(S, X_1) = C_{sx1} + f^*_2(x_1)$				$f^*_1(s)$	x^*_1
	S_{11}	S_{12}	S_{13}	S_{14}		
S_0	66+18	69+15	63+23	68+27		

Sumber : hasil penelitian (2014)

Tabel 15. Hasil Perhitungan Tahapan ke-1

X_1 / S	$f_1(S, X_1) = C_{sx1} + f^*_2(x_1)$				$f^*_1(s)$	x^*_1
	S_{11}	S_{12}	S_{13}	S_{14}		
S_0	84	84	86	95	84	S_{11} atau S_{12}

Hasil dari tahapan ke-1 maka diperoleh nilai $f^*_1(s)$ paling minimum adalah yang melalui S_{11} atau S_{12} dengan nilai $f^*1(s) = 84$. Dengan demikian pada tahapan ini yang diambil adalah rute S_{11} atau S_{12}

Dari ke-5 tahapan tersebut maka rute yang diperoleh adalah:

$S_o \rightarrow S_{11} \rightarrow S_{22} \rightarrow S_{33} \rightarrow S_{42} \rightarrow S_5$

Maka rute yang dilaluinya adalah dari kota 1 ke 2, 4, 5, 3 dan kembali lagi ke 1 dengan total lintasan = 99km.

D. Perbandingan Algoritma *Greedy* dengan Algoritma *Cheapest Insertion Heuristics*

Setelah diketahui jalur terpendek yang dihasilkan dengan algoritma *greedy* dan algoritma *cheapest insertion heuristics* dari kasus diatas yaitu 99 Km dengan jalur 1-3-5-4-2-1 maka dapat diperoleh perbandingan kedua algoritma tersebut seperti pada tabel 14.

Tabel 16. Perbandingan Algoritma

Algoritma	***Greedy***	***Cheapest Insertion Heuristics***	***Dinamik***
Kelebihan	- Pencarian lebih simple apabila kota tidak terlalu banyak - Dapat sampai tepat waktu menuju tujuan karena prinsip pencarian lintasan terpendek memakai fungsi seleksi dan itu berguna untuk menentukan jalan tersingkat untuk menuju suatu tempat.	-Algoritma ini masih stabil digunakan untuk kasus TSP dengan jumlah kota yang besar.	- Hasil perhitungannya optimal -Sangat tepat apabila diimplementasikan ke bahasa pemrograman pascal
Kekurangan	- Hanya mengambil solusi yang terbaik yang mereka temukan pada saat itu tanpa memperhatikan konsekuensi ke depan. -Keputusan optimal di setiap sub langkah tidak menghantarkan kita ke solusi optimal global yang diinginkan. - Kurang cocok digunakan apabila dengan jumlah kota yang besar	Prinsip pencariannya lebih rumit dan lebih lama dibanding algoritma *greedy* karena memungkinkan terjadinya perulangan perhitungan tambahan jarak (probabilitas edge yang akan digantikan) pada iterasi yang berbeda.	hasil perhitungan selanjutnya sangat bergantung dari hasil perhitungan sebelumnya sehingga apabila hasil perhitungan sebelumnya salah maka hasil selanjutnya pun salah

V. KESIMPULAN

Kesimpulan dari *paper* ini adalah:

1. Algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynami programming* adalah algoritma yang bisa digunakan untuk pemecahan TSP.
2. Dari hasil penyelesaian dengan algoritma *greedy*, algoritma *cheapest insertion heuristics* dan *dynamic programming* maka rute terpendek adalah 99 Km dengan jalur 1-3-5-4-2-1
3. Untuk kasus pada *paper* ini algoritma *greedy* lebih simple cara penyelesaiannya karena jumlah kotanya masih sedikit akan tetapi apabila jumlah kotanya lebih banyak maka lebih baik menggunakan algoritma *cheapest insertion heuristics.*
4. Ketiga algoritma tersebut memiliki kelebihan dan kekurangan masing-masing.
5. Jumlah kota yang dikunjungi mempengaruhi pemilihan algoritma.

DAFTAR PUSTAKA

Dian. 2013. Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem (Optimization Algorithm For Solving Travelling Salesman Problem*)*. Jurnal Transformatika. Jurusan Teknologi Informasi Fakultas Teknologi Informasi dan Komunikasi, Universitas Semarang. Volume 11, No.1, Juli 2013.

Dian, dkk. 2008. Evaluasi Kinerja Algoritma Travelling Salesman Problem Dengan Teknik Pemrograman Dinamik. Proceeding, Seminar Ilmiah Nasional Komputer dan Sistem Intelijen (KOMMIT 2008) Auditorium Universitas Gunadarma, Depok, 20-21 Agustus 2008

Kusrini dan Istiyanto, J.E.. 2007. *Penyelesaian Travelling Salesman Problem dengan Algoritma Cheapest Insertion Heuristic dan Basis Data*. *Jurnal Informatika*. Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra, Vol. 8, No. 2, Nopember 2007, 109-114.

Rusdi dan Siti. 2010. Studi Perbandingan Algoritma Cheapest Insertion Heuristic dan Ant Colony System Dalam Pemecahan Travelling Salesman Problem. Seminar Nasional Aplikasi Teknologi Informasi 2010 (SNATI 2010). Yogyakarta, 19 Juni 2010.

Winston, Wayne L. dan Goldberg, Jeffrey B. 2004. Operations Research Application And Algorithms 4th Edition. United States of America : Duxbury