PENYELESAIAN MASALAH *MINIMUM SPANNING TREE* (MST) MENGGUNAKAN *ANT COLONY SYSTEM* (ACS)

Irawan Afrianto[1], Euis Widiani Jamilah[2]

[1,2] Program Studi Teknik Informatika UNIKOM
Jl. Dipati Ukur 112-114 Bandung
E-mail : irawan@unikom.ac.id

ABSTRAK

Masalah optimasi jaringan menjadikan adanya kebutuhan untuk mencari nilai terkecil (minimal) pada suatu keadaan jaringan . Salah satu masalah optimasi jaringan adalah *Minimum spanning tree (MST)*, yaitu suatu keadaan dimana semua node dalam graf terhubung, namun tidak boleh terdapat loop didalamnya dan dihitung bobot tree yang terkecil. .Salah satu aplikasi MST adalah pembuatan jaringan komunikasi atau telepon yang akan menghubungkan semua stasiun telepon pada suatu kota yang ada. Permasalahannya adalah mencari jarak terpendek antara kota-kota tersebut sehingga penggunaan kabel akan lebih sedikit yang berarti menghemat biaya pembangunan jaringan telepon tersebut.

Ant Colony System (ACS) adalah salah satu algoritma heuristik yang dapat digunakan untuk menyelesaikan masalah MST tersebut. Dalam pencarian solusi kasus MST, Permasalahan direpresentasi seperti sekumpulan semut yang bekerja sama untuk menetukan solusi MST yang paling baik, semut-semut bekerja sama melalui komunikasi tidak langsung dengan menggunakan jejak pheromone yang disimpan pada sisi-sisi dari graph MST.

Dari hasil uji yang dilakukan dengan menggunakan parameter $\alpha = 1, \beta = 0.5, \tau = 0.1$, $\rho = 0.5, Q$ = 100, dan NC_{max} = 10 diperoleh kesimpulan bahwa ACS dapat dijadikan sebagai alternatif untuk memecahkan masalah MST meskipun tidak selalu memberikan solusi yang optimal, dikarenakan cara kerja ACS yang bersifat heuristik.

Kata kunci : Optimasi, *Minimum spanning tree*, Ant Colony System, Heuristik.

1. PENDAHULUAN

Kemajuan Teknologi komunikasi berkembang pesat seiring dengan meningkatnya kebutuhan informasi. Komunikasi adalah salah satu sarana untuk mendapatkan informasi dengan cepat, sebagai contohnya adalah telepon. Dengan telepon, manusia tidak lagi memerlukan waktu yang lama untuk menyampaikan informasi walaupun berbeda kota, propinsi dan negara. Tetapi masih ada kota yang belum bisa terjangkau dengan jaringan telepon, sehingga pendistribusian jaringan telepon tersebut belum optimum. Suatu jaringan yang optimum harus mampu menghubungkan semua kota atau semua tempat agar dapat berkomunikasi, dengan jarak sependek mungkin sehingga akan meminimumkan penggunaan kabel.

Masalah optimasi jaringan ini bisa diselesaikan salah satunya dengan menggunakan *Minimum spanning tree (MST). Minimum spanning tree* merupakan suatu graph yang memiliki bobot terkecil yang membentuk pohon (*Tree). Dengan Minimum spanning tree* ini, dapat merencanakan jaringan komunikasi atau distribusi dari objek yang dimiliki secara ekonomis, karena dicari jarak terpendek dari sejumlah stasiun jaringan telepon. Untuk menyelesaikan masalah tersebut, dapat digunakan beberapa metode atau algoritma, antara lain adalah algoritma *ant colony system* dan algoritma kruskal.

Algoritma *ant colony system* merupakan salah satu algoritma pencarian *heuristic.* Pada dasarnya *ant colony system* mengadaptasi cara kerja semut riil dalam pencarian makanan dengan mencari lintasan terpendek untuk sampai ke tujuan, *ant colony system* ini diharapkan dapat dijadikan sebagai alternatif untuk memecahkan masalah *Minimum spanning tree*. Untuk melihat performansi dari *ant colony system* maka digunakan algoritma kruskal sebagai algoritma pembanding. Algoritma Kruskal merupakan salah satu algoritma untuk pemecahan masalah *Minimum spanning tree* secara konvensional

2. TEORI PENDUKUNG

2.1 Graph

Graph merupakan representasi dari suatu masalah yang digambarkan sebagai sekumpulan noktah atau simpul (vertex) yang dihubungkan dengan sekumpulan garis atau sisi (edge). Secara singkat suatu graph dapat ditulis sebagai G = (V,E) yang dalam hal ini:

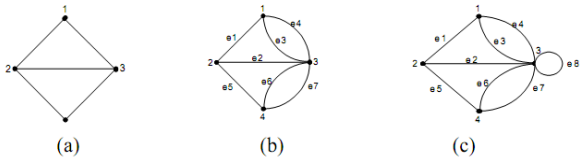
V = Himpunan berhingga dan tidak kosong dari simpul-simpul (vertex).

= { v1, v2,…vn }

E =Himpunan sisi yang menghubungkan sepasang simpul.

= { e1, e2,…en }

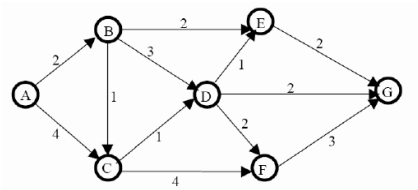
Simpul pada graph dapat dinomori dengan huruf, seperti v,w,…, dengan bilangan asli 1, 2,3,…, atau gabungan keduanya. Sisi (edge) yang menghubungkan dua simpul (vertex) vi dan vj dinyatakan dengan pasangan (vi , vj) atau dengan lambang e1, e2 , dengan kata lain, jika e adalah sisi yang menghubungkan simpul vi dan vj, maka e dapat ditulis sebagai e = (vi , vj).



Gambar 1 Graph Sederhana (a), Graph Ganda (b), Graph Semu (c)

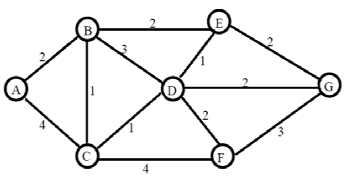
Menurut arah dan bobotnya, graf dibagi menjadi empat bagian, yaitu:

a. Graf berarah dan berbobot : tiap busur mempunyai anak panah dan bobot. Gambar 2. menunjukkan graf berarah dan berbobot yang terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik menujukkan arah ke titik B dan titik C, titik B menunjukkan arah ke titik D dan titik C, dan seterusnya. Bobot antar titik A dan titik B pun telah di ketahui.



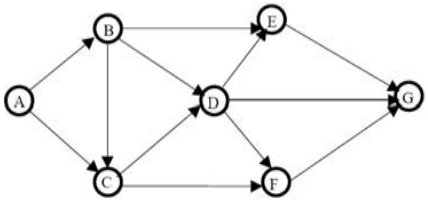
Gambar 2. Graf berarah dan berbobot

b. Graf tidak berarah dan berbobot : tiap busur tidak mempunyai anak panah tetapi mempunyai bobot. Gambar 3 menunjukkan graf tidak berarah dan berbobot. Graf terdiri dari tujuh titik yaitu titik A,B,C,D,E,F,G. Titik A tidak menunjukkan arah ke titik B atau C, namun bobot antara titik A dan titik B telah diketahui. Begitu juga dengan titik yang lain.



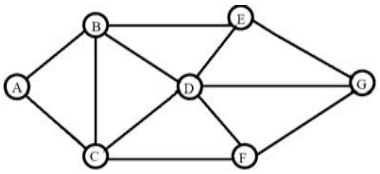
Gambar 3 Graf tidak berarah dan berbobot

c. Graf berarah dan tidak berbobot: tiap busur mempunyai anak panah yang tidak berbobot. Gambar 4 menunjukkan graf berarah dan tidak berbobot.



Gambar 4. Graf berarah dan tidak berbobot

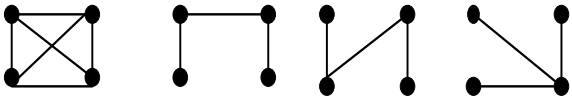
d. Graf tidak berarah dan tidak berbobot: tiap busur tidak mempunyai anak panah dan tidak berbobot.



Gambar 5 Graf tidak berarah dan tidak berbobot

2.2 Pohon Rentang (*Spanning Tree*)

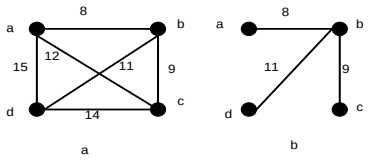
Pohon rentang merupakan suatu Subgraph dari suatu graph dimana setiap simpul pada pohon rentang sama dengan semua simpul pada graph. Gambar 6 adalah graph lengkap dan empat buah pohon rentangnya.



Gambar 6 Graph Lengkap G dengan pohon rentangnya T1, T2, T3

2.3 Pohon Rentang Minimum (*Minimum spanning tree*)

Pohon Rentang (Spanning Tree) pada suatu graph adalah subgraph minimal yang menghubungkan semua simpul pada graph. Apabila graph tersebut adalah graph berbobot (Weighted Graph), kemudian dari pohon rentang yang dimiliki oleh graph didefinisikan sebagai penjumlahan dari bobot – bobot seluruh cabang pada pohon rentang maka akan diperoleh pohon rentang yang memiliki bobot. Pohon rentang yang memiliki bobot terkecil pada suatu graph berbobot disebut Pohon rentang Minimum (*Minimum spanning tree*).

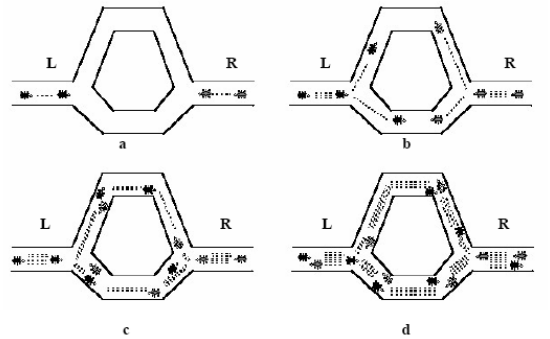


Gambar 7 (a) Graph berbobot dengan (b) pohon rentang minimumnya.

Secara umum penyelesaian masalah *Minimum spanning tree* dapat dilakukan dengan menggunakan metode konvensial dan metode heuristik. Metode konvensional menggunakan perhitungan matematis biasa untuk menyelesaikan masalah MST sepeti algortima Prim dan Kruskal. Sementara metode heuristik menggunakan kecerdasan buatan untuk menentukan rute terpendek pada MST, salah satu metode yang dapat digunakan adalah Ant Colony System (ACS).

2.4 Konsep *Ant Colony System (ACS)*

Koloni semut merupakan algorima yang bersifat heuristik untuk menyelesaikan masalah optimasi. Algoritma ini diinspirasikan oleh lingkungan kooni semut pada saat mencari makanan. Semut dapat mencari makanan. Semut dapat mencari lintasan terpendek dari suatu sumber makanan menuju sarangnya, tanpa harus melihatnya secara langsung. Karena terinspirasi dari semut asli dinamakan algoritma koloni semut. Semut-semut mempunyai penyelesaian yang unik dan sangat maju, yaitu menggunakan jejak pheromone pada suatu jalur untuk berkomunikasi dan membangun solusi, semakin banyak jejak pheromone ditinggalkan, maka jalur tersebut akan diikuti oleh semut lain.



Gambar 8. Ilustrasi semut menemukan sumber makanan

2.5 Penerapan Ant Colony System (ACS) Pada *Minimum spanning tree*

Ant Colony System (ACS) merupakan salah satu algoritma pencarian dalam penyelesaian kasus kombinatorial. Dalam pencarian solusi kasus kombinatorial, algoritma ini memanfaatkan informasi lokal dari solusi yang telah ditemukan sebelumnya secara *random* (acak) dan memperbaiki solusi yang telah ditemukan melalui sebuah mekanisme pengumpulan informasi.

Adapun langkah-langkah yang dilakukan oleh ACS untuk menyelesaiakan masalah *Minimum spanning tree* adalah sebagai berikut :

Langkah 1 : Inisialisasi harga parameter-parameter algoritma dan Inisialisasi kota pertama setiap semut.

Masukan algoritma adalah n (banyak kota) berikut x dan y (koordinat) atau d (jarak antar kota). Adapun Beberapa Parameter algoritma, yaitu :

1. Q : Tetapan Siklus Semut
2. α :Tetapan Pengendali Intensitas jejak semut
3. β : Tetapan Pengendali Visibilitas
4. η : Visibilitas antar kota = $1/d_{ij}$
5. m : Banyak semut
6. ρ : Tetapan penguapan jejak semut
7. NC_{max} : Jumlah siklus *maximum*, Bersifat tetap selama algoritma dijalankan
8. τ_{ij} : Intensitas jejak semut antar kota, akan selalu diperbaharui harganya pada setiap siklus algoritma mulai dari siklus pertama (NC = 1) sampai tercapai.

Jumlah siklus *maximum* (NC = NC_{max})atau sampai terjadi konvergensi. *Intensitas jejak semut antar kota* perlu diinisialisasi dengan harga tertentu. Biasanya harga yang digunakan adalah bilangan positif kecil dengan perubahan intensitas jejak semut antar kota sama dengan *nol*.

Misalkan b : t(i = 1,…,n) adalah jumlah semut didalam suatu kota *i* pada saat *t* dan misalkan $m = \sum_{i=1}^{n} b_i(t)$ adalah jumlah total semut. Setiap semut adalah kumpulan yang sederhana dengan karakteristik berikut :

1. Semut memilih kota yang akan dikunjungi dengan *probabilitas* yang difungsikan pada jarak kota dan jumlah jejak yang terdapat pada setiap hubungan batas.
2. Sebuah Struktur data, disebut *tabu list* yang diasosiasikan dengan tiap semut. *Tabu list* ini digunakan untuk menyimpan daftar kota yang telah dikunjungi dan larangan semut untuk mengunjunginya lagi sampai perjalanan selesai.
3. Saat perjalanan selesai, semut ini meninggalkan bagian yang disebut *trail* atau jejak pada setiap batas (*i*,*j*) yang telah dikunjungi.

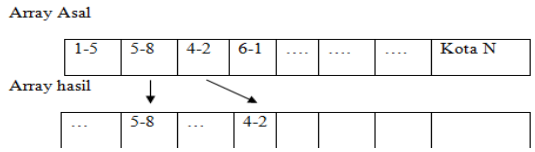
Algoritma dimulai dengan mendistribusikan semut pada kota pertama masing-masing. Setiap semut ditempatkan di kota-kota tertentu secara acak. Pendistribusian semut pada kota pertama inilah yang disebut dengan inisialisasi kota pertama setiap semut.

Langkah 2 : Pengisian kota pertama kedalam tabu list.

Kota-kota pertama setiap semut hasil inisialisasi langkah 1 harus diisikan sebagai elemen pertama

tabu list. Dari langkah ini akan didapat elemen pertama *tabu list* setiap semut berisi indeks kota tertentu. Jadi setiap ***tabu$_k$(l)*** bisa berisi indeks kota antara *l* sampai *n* sebagaimana hasil inisialisasi pada langkah 1.

Tabu list digunakan untuk menyimpan daftar urutan node-node yang sudah dikunjungi setiap semut. Pada awalnya, ketika algoritma belum dijalankan atau sebelum sebuah siklus kunjungan dimulai, *tabu list* setiap semut kosong. Kemudian setiap kali seekor semut berkunjung kesetiap kota, elemen *tabu list* bertambah satu, demikian seterusnya sampai *tabu list* penuh atau mempunyai banyak elemen sama dengan banyak semua node yang harus dikunjungi. Selain berfungsi sebagai penyimpan daftar urutan node, *tabu list* juga dipakai untuk mengetahui node-node yang belum dikunjungi semut sebelum sampai ke node terakhir.



Gambar 9. Array MST pada Tabu list

Langkah 3 : Penyusunan rute kunjungan setiap semut ke setiap kota.

Setelah semut-semut terdistribusi ke setiap node, semut-semut akan mulai melakukan perjalanan dari node pertama sebagai node asal dan salah satu node-node lainnya sebagai node tujuan. Kemudian dari node kedua semut-semut akan melanjutkan perjalanan dengan memilih salah satu dari node-node yang tidak terdapat pada ***tabu$_k$*** sebagai node tujuan berikutnya. Demikian seterusnya sampai semua node satu persatu telah menempati ***tabu$_k$***. Jika *s* menyatakan indeks urutan kunjungan, node asal dinyatakan sebagai sebagai ***tabu$_k$(s)*** dan node-node lainnya dinyatakan sebagai {*N*-**tabu$_k$**}, *i* merupakan node asal dan *j* menyatakan node tujuan, maka untuk menentukan node tujuan digunakan persamaan *probabilitas* node untuk dikunjungi sebagai berikut :

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{k' \in \{N - tabu_k\}} [\tau_{ik'}]^{\alpha}.[\eta_{ik'}]^{\beta}} \quad \text{,untuk j} \in \{\text{N - tabu}_k\}$$

$$p_{ij}^{k} = 0 \text{ , untuk j lainnya} \qquad (1)$$

Langkah 4 : Perhitungan Panjang Rute setiap Semut, Pencarian Rute Terpendek dan Perhitungan Perubahan Harga Intensitas Jejak Kaki Semut antar Node

Setelah satu siklus diselesaikan oleh semua semut, maka panjang rute atau L_k setiap semut dapat dihitung berdasarkan ***tabu$_k$*** masing-masing dengan persamaan berikut:

$$L_k = \sum_{s=1}^{n-1} d_{tabu_k(s).(s+1)} \qquad (2)$$

Dengan d_{ij} adalah jarak antara node *i* ke node *j* yang dihitung berdasarkan persamaan :

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \qquad (3)$$

Selanjutnya dicari harga minimal panjang rute setiap siklus atau L_{minNC} sebagai :

L_{minNC}= min (L_1,L_2, ...L_m) (4)

Sedangkan harga minimal panjang rute secara keseluruhan adalah

L_{min} = min (L_{min1}, $L_{min2,....,}L_{min3}$) (5)

Perjalanan semut antar node akan meninggalkan jejak-jejak kaki pada semua lintasan yang dilaluinya. Adanya 'penguapan' menyebabkan kemungkinan terjadinya perubahan harga intensitas jejak kaki semut antar node. Persamaan perubahan ini adalah:

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau^{k}{}_{ij} \qquad (6)$$

Dengan $\Delta\tau^{k}_{ij}$ adalah perubahan harga intensitas jejak kaki semut antar node setiap yang dihitung dengan persamaan sebagai berikut :

$\Delta\tau^{k}{}_{ij} = \frac{Q}{L_k}$, untuk (i,j) ε node asal dan node tujuan dalam ***tabu$_k$***

$\Delta\tau_{ij} = 0$, untuk (i,j) lainnya (7)

Langkah 5 : Perhitungan Harga Intensitas Jejak kaki Semut antar Node untuk Siklus Berikutnya dan Reset Harga intensitas Jejak kaki semut antar node

Semua lintasan antar node yang dijadikan jalur semut mempunyai kemungkinan untuk dilewati semut-semut pada siklus berikutnya. Karena adanya 'penguapan' dan intensitasnya juga mengalami perubahan tergantung pada semut-semut yang melewatinya, maka dimata semut yang akan lewat pada lintasan tersebut untuk siklus berikutnya, harga intensitas sudah berubah. Harga Intensitas jejak kaki semut antar node untuk siklus berikutnya dihitung dengan persamaan :

$$\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij}$$

Selanjutnya untuk siklus yang berikutnya perubahan harga intensitas jejak semut antar node perlu di-reset kembali agar berharga sama dengan nol.

Langkah 6 : Pengosongan tabu list

Apabila belum tercapai jumlah siklus maksimum atau belum terkonvergensi, maka algoritma perlu diulang lagi dari langkah 2 dengan harga parameter intensitas jejak kaki semut antar node yang sudah diperbaharui. Disamping itu tabu list perlu dikosongkan untuk diisi lagi dengan urutan node yang baru pada siklus berikutnya. Proses ini berlangsung sampai perjalanan mencapai jumlah maksimum pada alur NC_{max} atau semua semut membuat pola perjalanan yang sama. Karena hal ini

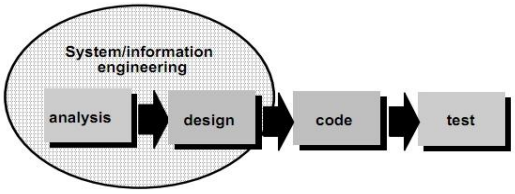
disebut dengan *Stagnation Behavior* karena ini digunakan pada situasi dimana titik algoritma mencapai solusi alternatif.

3. PEMBAHASAN

3.1 Metode Penelitian

Metode penelitian yang dilakukan dalam penelitian ini mencakup kegiatan :

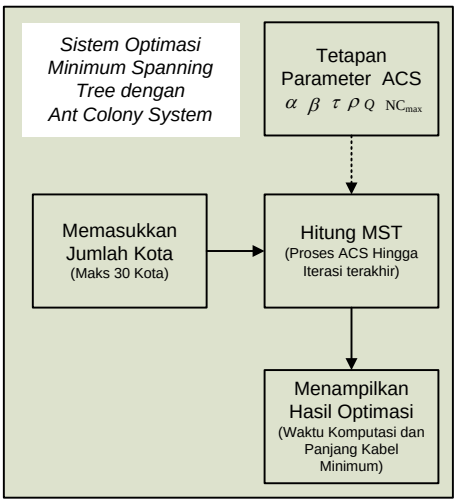
a. Metode kepustakaan
   Mengumpulkan sumber-sumber dari buku-buku teks, jurnal dan penelitian-penelitian yang relevan dengan penelitian
b. Pengembangan Perangkat Lunak
   Menggunakan pendekatan waterfall dalam pengembangan perangkat lunak Ant Colony System dalam penyelesaian MST.



Gambar 10. Model Waterfall – Pressman

3.2 Anasisis Sistem

Sistem bekerja dengan memasukkan jumlah kota yang akan dihitung *Minimum spanning tree*nya (dalam hal ini panjang kabel minimum). Tetapan parameter ACS digunakan sebagai parameter masukan guna mengeksekusi algoritma ACS dalam memproses masalah MST sesuai dengan alur kerja algoritmanya. Setelah proses komputasi ACS-MST selesai dilakukan (berhenti karena iterasi maksimum), maka sistem akan mengeluarkan keluaran berupa waktu komputasi dan panjang kabel minimum yang menghubungan antar kota tersebut.



Gambar 11. Alur Kerja Sistem ACS-MST

3.3 Analisis Data dan Parameter ACS

Data yang digunakan dalam penelitian ini adalah data jarak antar kota di Jawa Barat, yang dibuat dalam bentuk tabel jarak kota seperti pada Tabel 1. Jumlah kota dalam penelitian ini dibatasi sebanyak 30 kota.

Tabel 1. Data Jarak Antar Kota

	1	2	3	4	5	...	30
1	0	15.27	38.81	54.41	58.97	73.95	251.07
2	15.27	0	24.72	42.94	48.62	61.8	236.22
3	38.81	24.72	0	44.2	51.51	58.2	218.69
4	54.41	42.94	44.2	0	7.41	19.69	201.89
5	58.97	48.62	51.51	7.41	0	17.63	201.19
...	73.95	61.8	58.2	19.69	17.63	0	183.56
30	251.07	236.2	218.69	201.89	210.19	183.56	0

Keterangan :
1 : Pandeglang , 2 : Rangkasbitung, 3.: Cipanas,
4 : Tangerang, 5 : Bandara Soekarno Hatta,…., 30 : Cirebon.

Data tersebut direpresentasikan dalam bentuk array 2 dimensi dan merupakan graf lengkap, karena semua kota terhubung dengan semua kota lainnya. Sementara untuk parameter ACS yang digunakan untuk penyelesaian masalah MST dapat dilihat pada Tabel 2.

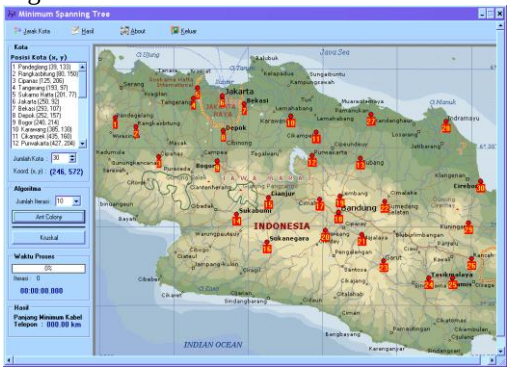
Tabel 2. Parameter ACS

Parameter	Nilai
α	1
β	5
τ	0.1
ρ	0.5
Q	100
NC_{max}	10

4. IMPLEMENTASI dan PENGUJIAN SISTEM

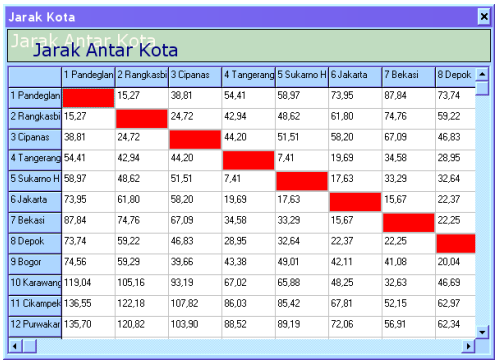
4.1 Implementasi Sistem

Aplikasi penyelesaian masalah MST dengan ACS ini dikembangkan menggunakan Borland Delphi 7.0 dengan maksimal kota yang dapat dimasukkan sebanyak 30. Didalam aplikasi tersebut menyertakan algoritma konvensional (algoritma Kruskal) sebagai pembanding algortima ACS. Adapun tampilan dari aplikasi MST – ACS terlihat pada gambar 12.



Gambar 12. Tampilan Aplikasi MST – ACS

Adapun jarak yang akan dihitung pada MST adalah jarak antar kota di Jawa Barat yang dimodelkan berupa graf lengkap, dimana setiap kota terhubung ke kota lainnya, seperti pada gambar 13.



	1 Pandeglan	2 Rangkasbi	3 Cipanas	4 Tangerang	5 Sukarno H	6 Jakarta	7 Bekasi	8 Depok
1 Pandeglan		15,27	38,81	54,41	58,97	73,95	87,84	73,74
2 Rangkasbi	15,27		24,72	42,94	48,62	61,80	74,76	59,22
3 Cipanas	38,81	24,72		44,20	51,51	58,20	67,09	46,83
4 Tangerang	54,41	42,94	44,20		7,41	19,69	34,58	28,95
5 Sukarno H	58,97	48,62	51,51	7,41		17,63	33,29	32,64
6 Jakarta	73,95	61,80	58,20	19,69	17,63		15,67	22,37
7 Bekasi	87,84	74,76	67,09	34,58	33,29	15,67		22,25
8 Depok	73,74	59,22	46,83	28,95	32,64	22,37	22,25	
9 Bogor	74,56	59,29	39,66	43,38	49,01	42,11	41,08	20,04
10 Karawang	119,04	105,16	93,19	67,02	65,88	48,25	32,63	46,69
11 Cikampek	136,55	122,18	107,82	86,03	85,42	67,81	52,15	62,97
12 Purwakar	135,70	120,82	103,90	88,52	89,19	72,06	56,91	62,34

Gambar 13. Tampilan Jarak Antar Kota MST – ACS

Keluaran dari aplikasi MST-ACS adalah berupa simulasi *Minimum spanning tree* yang dihasilkan yang menghubungkan jumlah kota sesuai pilihan.

Gambar 14. Tampilan Keluaran MST – ACS

4.2 Pengujian Sistem

Sistem diuji dengan menggunakan parameter ACS yang telah ditetapkan pada Tabel 2 dengan jumlah kota yang akan diuji adalah 8, 10, 15, 20, 25, dan 30 kota. Adapun hasil keluaran sistem adalah MST yang diperoleh (panjang kabel) serta waktu komputasinya.

Tabel 3. Hasil Pengujian Sistem

Jumlah Kota	Panjang Kabel (km)		Waktu Komputasi (ms)	
	ACS	Kruskal	ACS	Kruskal
8	145,88	145,88	108	547
10	195,27	195,27	187	438
15	324,40	324,40	640	719
20	412,68	412,68	2219	2719
25	531,60	531,60	8313	2000
30	674,78	674,78	14219	3063

Hasil pengujian menunjukkan bahwa algoritma ACS mampu menghasilkan *Minimum spanning tree* yang sama dengan algoritma kruskal. Waktu komputasi algortima ACS lebih cepat untuk jumlah kota dibawah 20 kota, sementara untuk jumlah kota diatas 25 kota, algoritma Kruskal lebih cepat proses komputasinya.

5. PENUTUP

Dari hasil pengujian yang dilakukan dapat diambil kesimpulan sebagai berikut :

1. Ant Colony System dapat dijadikan sebagai alternatif untuk memecahkan masalah *Minimum spanning tree*, hasil yang diperoleh sama algoritma konvensioanl (kruskal) dalam penyelesaian masalah *Minimum spanning tree*.
2. Susunan kota yang dihubungkan untuk menghasilkan panjang kabel minimum setiap algoritma berbeda, namun menghasilkan jarak MST yang sama.
3. Pada Algoritma ant colony system, iterasi sangat berpengaruh pada waktu komputasi, walaupun susunan stasiun telepon yang terlewati relatif sama.
4. Dilihat dari segi waktu komputasi : Pada algoritma semut untuk iterasi 10, waktu komputasi lebih cepat jika jumlah stasiun telepon yang dihubungkan kurang dari 15. Sedangkan jika iterasinya 25, proses lebih cepat jika stasiun yang dihubungkannya kurang dari 11. Pada Algoritma Kruskal, Proses komputasi lebih cepat jika stasiun telepon yang dihubungkannya lebih dari 15.

DAFTAR PUSTAKA

[1]. Afrianto, Irawan. ***Perbandingan Algoritma Konvensional dan Algoritma Genetik dalam memecahkan masalah Minimum spanning tree,*** Skripsi, Jurusan Teknik Informatika, UNIKOM, 2002.

[2]. Deo, Narsingh. ***Graph Theory (With aplications To Engineering And Computer Science),*** Prentice-Hall of India Pvt.Ltd, 2004.

[3]. Dimyati, Tjuju T,. Dimyati ,Ahmad. ***Operations Research***, Sinar baru Algensindo, 2002.

[4]. Kusumadewi, Sri,. Purnomo, Hari,. ***Penyelesaian Masalah Optimasi dengan Teknik – teknik Heuristik***, Yogyakarta: Graha Ilmu, 2005.

[5]. Lipschutz, Saymour. Lipson, Marc Lars. ***Matematika Dis.krit***. Jilid 2. Penerbit Salemba Teknika, 2002.

[6]. Marco Dorigo dan Gianni Di Caro, ***The Ant Colony Optimization Meta – Heuristic*** [1]. Belgium, 1996.

[7]. Sufa'atin, ***Penyelesaian Travelling Salesman menggunakan Simulated Annealing***, Skripsi; Jurusan Teknik Informatika, UNIKOM, 2003.