ALGORITMA GENETIC ANT COLONY SYSTEM UNTUK MENYELESAIKAN TRAVELING SALESMAN PROBLEM

Lutfiani Safitri [1] **Sri Mardiyati** [2]

[1] Matematika, FMIPA Universitas Indonesia
Jl. H. Boan lisan 9, Depok 16425 Indonesia
email : [1]Lutfiani.safitri@gmail.com, [2]Srimardiyati25@gmail.com
[2] Matematika, FMIPA Universitas Indonesia
Jl. Bandung 4, Menteng Jakarta Pusat 101310 Indonesia
email :

ABSTRACT

Traveling Salesman Problem (TSP) is an optimization problem in which a person will travel to a number of cities, starting from origin city to visit every city exactly once and return to origin city with minimum total cost or distance. In this paper, GACS algorithm will be used to solve TSP. Then the results will be compare with ACS algorithm. And the conclution,GACS algorithm better than ACS algorithm.

Key words

Traveling Salesman Problem (TSP), Ant Colony System (ACS), Genetic Algorithm, Genetic Ant Colony System (GACS).

1. Pendahuluan

Traveling salesman problem (TSP) merupakan permasalahan dimana seseorang akan melakukan perjalanan kesejumlah kota yang dimulai dari kota asal dengan mengunjungi semua kota tepat satu kali dan kembali ke kota asal dengan total biaya atau jarak perjalanannya minimal [9]. TSP dapat pula dikatakan sebagai permasalahan kombinatorial dengan kemungkinan penyelesaian yang banyak, serta penyelesaian yang akan diambil adalah penyelesaian yang terbaik [5]. Pada proses penyelesaiannya, TSP ini dinyatakan dalam suatu graf lengkap. Banyak permasalahan yang dapat direpresentasikan dalam bentuk TSP, diantaranya adalah pencarian rute bus sekolah, efisiensi pengiriman surat atau barang, dan perancangan pemasangan pipa saluran [1].

Permasalahan TSP dibagi dalam dua tipe, yaitu *Asymmetric Traveling Salesman Problem* (ATSP) dan *Symmetric Traveling Salesman Problem* (STSP). *Symmetric Traveling Salesman Problem* ini biasa disebut sebagai *Traveling Salesman Problem* (TSP) saja [8]. Ada dua metode yang dapat digunakan untuk menyelesaikan permasalahan TSP, yaitu metode eksak dan metode heuristik. Beberapa metode eksak yang dapat digunakan untuk menyelesaikan TSP adalah metode *cutting-plane* [Ralph Gomory, 1950], Algoritma *Branch and bound* [A.H Land, 1960], Algoritma *Held-karp* [Bellman, 1962], dan lain-lain. Metode eksak ini dapat menghasilkan solusi yang optimal, namun tidak mungkin digunakan untuk menyelesaikan permasalahan dengan jumlah simpul yang banyak, karena akan membutuhkan waktu yang sangat lama. Oleh karena itu berkembanglah metode-metode heuristik yang dapat digunakan untuk menyelesaikan permasalahan dengan jumlah simpul yang besar, walaupun metode ini hanya menghasilkan solusi pendekatan. Beberapa metode heuristik yang dapat digunakan untuk menyelesaikan TSP adalah *V-opt Method* [Shen Lin, 1972], *Simulated annealing* [Krikpatrick, 1983], *Genetic Algorithm* [John Holland, 1970], *Ant Colony Optimization* [Marco Dorigo, 1992], dan lain-lain [8].

Ant Colony Optimization (ACO) adalah kumpulan algoritma untuk menyelesaikan masalah optimasi kombinatorial, yang terinspirasi dari perilaku sosial semut. Pada kehidupan nyata, semut-semut akan meninggalkan sarangnya untuk mencari rute menuju sumber makanan dan akan kembali lagi kesarangnya. Dalam proses pencarian rute tersebut, semut akan mengeluarkan zat kimia yang disebut *pheromone* untuk menandai rute yang telah dilewatinya dan rute yang telah dilewati oleh semut tersebut disebut sebagai jalur *pheromone* . Proses pembentukan jalur tersebut yang menentukan semut dapat memilih rute terbaik yang akan dilalui dari sarang menuju sumber makanan yang ditemukannya [7].

Algoritma pertama yang termasuk dalam ACO adalah algoritma *Ant System*, algoritma ini diusulkan pertama kali oleh Dorigo pada tahun 1997. Kemudian *Ant System*

diperbaharui oleh Stutzel T. Hoos pada tahun 1997 menjadi *Max-Min Ant System*. Algoritma *Max-Min Ant System* ini mengalami pembaharuan lagi menjadi algoritma *Ant Colony System* yang diusulkan oleh Dorigo pada tahun 1997. ACS ini merupakan algoritma terbaru dalam ACO, namun dalam pencarian solusinya ACS ini belum mendapatkan hasil yang terbaik [3]. Oleh karena itu dalam makalah ini akan dibahas mengenai ACS yang dikombinasikan dengan algoritma genetika untuk menyelesaikan masalah TSP, dengan tujuan mendapatkan solusi yang lebih baik dari sebelumnya.

2. Pemodelan

Pada Bab ini akan dijelaskan mengenai pemodelan dari *Traveling Salesman Problem*. Kemudian juga akan dijelaskan algoritma yang akan digunakan pada makalah ini.

2.1 Traveling Salesman Problem

Traveling Salesman Prolem (TSP) petama kali dipelajari pada abad ke-18 oleh seorang matematikawan asal Irlandia yang bernama Sir William Rowam Hamilton dan seorang matematikawan asal Inggris yang bernama Thomas Penyngton Kirkman [5]. Diskusi rinci tentang pekerjaan Hamilton dan Kirkman ini dikembangkan dalam sebuah buku yang berjudul *Graph Theory*. Sejak saat itu, masalah Hamilton dan Krikman dikenalkan sebagai *Travelling Salesman Problem* oleh Hassler, Whitney dan Merrill di Princeton [5].

Definisi 2.1

TSP merupakan permasalahan dimana seseorang akan melakukan perjalanan kesejumlah kota dalam satu perjalanan. Dimulai dari kota asal dengan mengunjungi semua kota tepat satu kali dan kembali ke kota asal dengan total biaya atau jarak perjalanannya minimal [9].

Dalam proses penyelesaiannya TSP ini dinyatakan dalam suatu graf lengkap G, dengan himpunan simpul V yang merepresentasikan *n* kota dan himpunan busur *E* yang merepresentasikan jalur perjalanan antar kota, dimana setiap busur memiliki bobot yang merepresentasikan biaya, jarak atau waktu $C(i,j)$ untuk setiap pasangan kota (i,j). Dalam makalah ini, parameter yang akan digunakan adalah jarak yang artinya bobot pada setiap busurnya merepresentasikan jarak antar kota [5].

Berdasarkan parameter jaraknya permasalahan TSP ini dibagi dalam dua kelompok, yaitu *Asymmetric Traveling Salesman Problem* (ATSP) dan *Symmetric Traveling Salesman Problem* atau yang biasa disebut TSP saja, dalam makalah ini TSP yang akan digunakan adalah *Symmetric Traveling Salesman Problem*. Pada STSP besarnya parameter jarak dari kota *i* ke kota *j* akan sama dengan arah sebaliknya, yaitu dari kota *j* ke kota *i*, yang artinya besar bobot antar kotanya simetris. Sehingga jika *d* merupakan jarak tempuh perjalanan antar kota, maka dapat dituliskan dengan $d(i,j) = d(j,i)$ [5].

Misalkan STSP digambarkan dengan sebuah graf lengkap tidak berarah $G = (V,E)$ dimana himpunan $V = \{1, 2, \dots, n\}$ adalah himpunan simpul yang merepresentasikan kota-kota yang akan dikunjungi, E adalah himpunan busur tak berarah yang merepresentasikan jalur perjalanan dari kota *i* ke kota *j* dan d_{ij} adalah bobot busur yang merepresentasikan jarak perjalanan antar kota *i* dan kota *j*. Kemudian didefinisikan x_{ij} merupakan variabel keputusan yang akan bernilai 1 ketika *salesman* melakukan perjalanan dari kota *i* ke kota *j* dan bernilai 0 untuk lainnya, maka formulasi dari STSP adalah

$$\min \sum_{i<j}^{n} d_{ij} x_{ij}$$

dengan kendala

$$\sum_{i<k} x_{ik} \sum_{j>k} x_{kj} = 2 \quad (k \in V) \tag{2.1}$$

$$\sum_{i \in S, j \in S, i \neq j} x_{ij} \leq |S| - 1, S \subset V, \quad 3 \leq |S| \leq n - \tag{2.2}$$

(Davendra,2010)

Kendala (2.1) menunjukkan bahwa setiap simpul akan terdapat dua busur yang terhubung dengan simpul tersebut, yang merepresentasikan bahwa setiap kota akan dikunjungi tepat satu kali dan akan mengunjungi kota lainnya tepat satu kali. Sedangkan kendala (2.2) digunakan untuk menghindari adanya *cycle* pada sebagian simpul yang merepresentasikan adanya *subtour* (sebagian rute perjalanan) yang membentuk rute perjalanan tertutup dengan memeriksa setiap subhimpunan S dari V mulai dari $|S| = 3$ sampai $|S| = n - 3$, dimana n adalah jumlah kota yang akan dikunjungi [5].

2.2 Algoritma *Ant Colony System*

ACO adalah kumpulan algoritma untuk menyelesaikan masalah optimisasi kombinatorial, pertama kali diperkenalkan oleh Marco Dorigo pada tahun 1992. Algoritma ini terinspirasi dari pengamatan langsung koloni semut. Dalam kehidupan nyata, semut bergerak secara acak mencari makanan untuk dibawa pulang ke sarangnya. Dengan membawa makanan, semut menandai jalur pulangnya dengan menjatuhkan hormon feromon. Apabila

semut-semut lain menemukan jalur ini, mereka akan mengikuti jalur ini dan ketika kembali dengan membawa makanan, mereka akan memperkuat jalur ini dengan menjatuhkan *pheromone* lagi. *Pheromone* dapat menguap dengan cepat. Ketika *pheromone* menguap, daya atraktifnya akan berkurang. Semakin lama waktu yang dibutuhkan oleh seekor semut untuk mengikuti jalur dan kembali lagi, semakin banyak feromon yang akan menguap. Begitu pula sebaliknya, sebuah jalur yang pendek, akan dilewati lebih cepat dan kadar feromon tetap tinggi, karena feromon terus menerus dijatuhkan oleh semut yang melewatinya [7].

Beberapa Algoritma yang termasuk dalam ACO adalah algoritma *Ant System*, algoritma ini diusulkan pertama kali oleh Dorigo pada tahun 1997. Kemudian *Ant System* diperbaharui oleh Stutzel T. Hoos pada tahun 1997 menjadi *Max-Min Ant System*. Algoritma *Max-Min Ant System* ini mengalami pembaharuan lagi menjadi algoritma *Ant Colony System* yang diusulkan oleh Dorigo pada tahun 1997 [3]. Algoritma ACS ini dapat diaplikasikan pada banyak masalah optimisasi kombinatorial, diantaranya adalah *Traveling Salesman Problem* (TSP), *General Assignment Problem* (GAP), *Vehicle Routing Problem* (VRP), dan *Quadratic Assignment Problem* (QAP) [6]. Pada makalah ini akan menyelesaikan TSP dengan menggunakan algoritma ACS yang akan dikombinasikan dengan algoritma genetika. Berikut adalah alur dari algoritma ACS:

- Cari panjang antar kota dan inisialisasi awal dengan menentukan nilai feromon awal, kota awal yang akan dikunjungi , dan menentukan parameter .
- Untuk setiap semut, pilih kota selanjutnya yang akan dilalui dengan menghitung probabilitas $p_k(r,s)$, dimana $p_k(r,s) = \begin{cases} \frac{\tau(r,s)^{\alpha}.\eta(r,s)^{\beta}}{\sum_{u \in j_r^k} \tau(r,u)^{\alpha}.\eta(r,u)^{\beta}}, & \text{jika } s \in j_r^k \\ 0, & \text{sebaliknya} \end{cases}$.
- Lakukan pembaharuan tingkat *pheromone* (pembaharuan lokal) pada setiap edge yang telah dilalui oleh semut, dengan formula $\tau(r,s) \leftarrow (1-\rho).\tau(r,s) + \rho.\tau_0$.
- Jika semua semut telah mengunjungi semua kota lanjutkan ke langkah berikutnya. Jika semua semut belum mengunjungi semua kota, ulangi langkah 2 dan 3 sampai semua semut mengunjungi semua kota tepat satu kali.
- Setelah semua kota telah dikujungi dan telah membentuk suatu rute, hitung total jarak yang dilalui oleh semut dalam membentuk sebuah rute.
- Dari semua rute yang telah dilewati oleh semut, pilih rute yang mempunyai total jarak tempuh terpendek. Kemudian lakukan pembaharuan untuk tingkat *pheromone* (pembaharuan global) pada edge yang termasuk dalam rute terpendek, dengan formula $\tau(r,s) \leftarrow (1-\rho).\tau(r,s) + \rho.\Delta_{\tau_k}(r,s)$, $\Delta_{\tau_k}(r,s) = \begin{cases} \frac{Q}{L_{gb}}, & \text{jika } (r,s) \in \text{jalur terpendek} \\ 0, & \text{lainnya} \end{cases}$.
- Lakukan pengulangan dari langkah 2 sampai langkah 6 sampai didapatkan hasil yang optimal.[3]

2.3 Algoritma Genetika

Algoritma Genetika pertama kali ditemukan oleh John Holland dari Universitas Michigan pada tahun 1960. Kemudian bersama dengan murid dan teman-temannya, John Holland mempublikasikan secara resmi Algoritma Genetika dalam sebuah buku yang berjudul *Adaptation of Natural and Artificial System* pada tahun 1975 [4].

Algoritma ini merupakan algoritma pencarian yang berdasarkan pada seleksi alam dan genetika alam, yang dikenal dengan proses evolusi [10]. Dalam proses evolusi, individu secara terus-menerus mengalami perubahan gen untuk menyesuaikan dengan lingkungan hidupnya melalui proses perkembangbiakan. Dalam algoritma genetika ini proses perkembangbiakan menjadi proses dasar yang menjadi perhatian utama, dengan dasar berpikir "Bagaimana cara mendapatkan keturunan yang lebih baik" [4].

Sistem Algoritma Genetika ini mengkodekan solusi kedalam bentuk numerik yang disebut kromosom. Elemen dasar kromosom disebut dengan gen. kemudian secara acak menjadikan sejumlah kromosom sebagai populasi awal. Lalu akan dipilih dua kromosom secara acak dari populasi untuk dilakukan operasi *Crossover* dan operasi mutasi, dilakukan berulang kali sampai hasilnya berhenti dalam kondisi yang optimal atau jumlah maksimum generasinya tercapai. Setelah melakukan operasi *crossover* dan operasi mutasi, sistem akan menghitung nilai fitness setiap kromosom. Kromosom dengan nilai fitness yang lebih tinggi akan dipilih ke dalam kolam gen untuk dilakukan reproduksi pada generasi berikutnya [3]. Berikut adalah prosedur dari proses algoritma genetika

- Bangun populasi awal.
- Hitung nilai fitness untuk setiap kromosom
- Lakukan seleksi dari populasi awal untuk dilakukan operasi *crossover*. Setelah terpilih kemudian bentuk pasangan antar dua kromosom yang akan menjadi kromosom orang tua.
- Setelah pemilihan kromosom orang tua, lakukan operasi *crossover* dari setiap pasang kromosom orang tua. Kemudian akan didapatkan kromosom keturunan.
- Lakukan pemilihan untuk kromosom yang akan dilakukan mutasi. Kemudian lakukan proses mutasi

pada kromosom yang terpilih, dan hitung nilai fitness dari kromosom tersebut.

- Membentuk populasi baru, yaitu dengan membandingkan nilai fitness dari kromosom keturunan dengan nilai fitness kromosom orang tuanya. Jika nilai fitness keturunan lebih bagus, maka ganti kromosom orang tua dengan kromosom keturunan yang baru. Jika tidak, maka tidak dilakukan penggantian.
- Kemudian, ulangi langkah 3 dan 4 untuk mendapatkan hasil yang optimal.[3]

3. Hasil Percobaan

Akan digunakan enam data set dari *Traveling Salesman Problem* yang masing-masing data berisi 5 kota, 6 kota, 7 kota, 20 kota, 70 kota dan 101 kota. Data berisi titik koordinat (x,y) yang menunjukkan letak suatu kota. Kemudian dari data tersebut akan dihitung jarak antar kotanya. Data tersebut akan dilakukan simulasi terhadap algoritma ACS dan algoritma GACS, yang kemudian hasilnya akan dibandingkan. Dalam penelitian ini akan digunakan beberapa parameter terbaik untuk permasalahan yang digunakan, dan berikut adalah parameternya [3]

Tabel 3.1 Parameter yang Digunakan Dalam Simulasi

Parameter	
β	*2*
α	*1*
ρ	*0.1*
Tingkat crossover	*1*
R_0	*0.33*
Tingkat mutasi rute (ϕ_r)	*0.3*
Tingkat mutasi pheromone (ϕ_p)	*0.2*

Dari data tersebut, dengan menggunakan parameter yang telah disebutkan pada tabel 3.1, kemudian akan dilakukan proses pencarian solusi sesuai dengan alur algoritma yang telah diimplementasikan dalam perangkat lunak tersebut. Output yang akan dikeluarkan dalam simulasi tersebut berupa total jarak perjalanan yang ditempuh oleh seorang sales, dan solusi yang optimum adalah solusi dengan total jarak yang paling minimum. Kemudian didapatkan hasil simulasinya yang akan disajikan pada tabel berikut.

Tabel 3.2 Hasil Simulasi Program

Jumlah Kota	***Algoritma***	***Jarak***	***Waktu Komputasi (Dalam Detik)***
5 Kota	*ACS*	*13.87*	*0.0014*
	GACS	*13.87*	*0.0014*
6 Kota	*ACS*	*164*	*0.015*
	GACS	*164*	*0.01499*
7 Kota	*ACS*	*166*	*0.03099*
	GACS	*166*	*0.015*
20 Kota	*ACS*	*39314*	*3.797*
	GACS	*33118*	*1.141*
70 Kota	*ACS*	*2359*	*1090.697*
	GACS	*1986*	*200.4*
101 Kota	*ACS*	*2412*	*4569.94*
	GACS	*2017*	*1297.41*

Dari tabel diatas terlihat bahwa untuk ukuran kota yang kecil (kurang dari 7) waktu komputasi yang dibutuhkan algoritma ACS dan algoritma GACS sama, dan hasil yang didapatkan juga sama. Pada data 7 kota didapatkan hasilnya sama, namun waktu komputasi yang dibutuhkan algoritma GACS lebih sedikit dibandingkan dengan waktu komputasi yang dibutuhkan algoritma ACS. Sedangkan untuk data yang besar terlihat bahwa hasil yang didapatkan oleh algoritma GACS lebih optimal dibandingkan dengan algoritma ACS dan waktu komputasi yang dibutuhkan juga lebih sedikit dibandingkan dengan algoritma ACS.

4. Kesimpulan

- Berdasarkan simulasi pada tabel 3.2, didapatkan bahwa algortima *Genetic Ant Colony System* (GACS) dapat diterapkan untuk menyelesaikan *Traveling Salesman Problem* (TSP).
- Berdasarkan hasil simulasi pada tabel 3.2, terlihat bahwa hasil penyelesaian TSP dengan menggunakan GACS lebih baik dibandingkan dengan menggunakan ACS
- Berdasarkan hasil simulasi pada tabel 3.2, untuk simpul ≥ 5 waktu komputasi pada algoritma GACS lebih cepat dibandingkan dengan waktu komputasi algoritma ACS.

REFERENSI

[1] Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2007). *The Traveling Salesman Problem.* New Jersey: Princenton University Press.

[2] Chartrand, G., & Zhang, P. (2005). *Introduction to Graph Theory.* Singapore: McGraw Hill.

[3] Chen, S.-M., & Chien, C.-Y. (2011). Parallelized Genetic Ant Colony System for Solving The Traveling Salesman Problem. *Expert System with Applications*, 3873-3883.

[4] Coley, D. A. (1999). An Introduction to Genetic Algorithms for Scientists and Engineers. USA: World Scientific.

[5] Davendra, D. (2010). Traveling Salesman Problem, Theory and Applications. India: IntechWeb.Org.

[6] Dorigo, M., & Gambardella, L. M. (1997). Ant Colonies for The Traveling Salesman Problem. *Biosystems*, 43,73-81.

[7] Dorigo, M., & Stutzel, T. (2004). *Ant Colony Optimization.* USA: Massachusetts Institute Thecnology.

[8] Gutin, G., & Punnen, A. (2000). *The Traveling Salesman Problem and Its Variations.* London: Kluwer Academic Publisher.

[9] Hillier, F. S., & Lieberman, G. J. (2001). *Introduction to Operations Research.* USA: McGraw Hill.

[10] Holland, J. H. (1975). *Adaptation in Natural and Artificial System.* Michigan: University of Michigan Press.

[11] Jacob, B. (1990). *Linear Algebra.* USA: W. H. Freeman and Company.

[12] Liu, X. H., Liu, W. J., & Jin, T. G. (2007). Improved MMAS or multi-process routes decision-making. *IEEE conference on industrial electronics and applications*, 1426-1430.

[13] Obitko, (1998). Dipetik April 2014, dari www.obitko.com/tutorials/genetic-algorithms.

[14] Ostfeld, A. (2011). Ant Colony Optimization-Methods and Applications . India: InTech.

[15] Yang, H., Li, X., Bo, C., & Shao, X. (2006). A graphic clustering algorithm based on MMAS. *IEEE congress on evolutionary computation*, 1592-1597.