Penggunaan Graf dalam Algoritma Semut untuk Melakukan Optimisasi

Ibnu Sina Wardy – NIM : 13505035

Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : if15035@students.if.itb.ac.id

Abstrak

Makalah ini membahas tentang penggunaan graf dalam algoritma semut untuk melakukan optimisasi. Algoritma semut atau *Ant Colony Optimization* merupakan sebuah algoritma yang berasal dari alam. Algoritma semut merupakan teknik probabilistik untuk menyelesaikan masalah komputasi dengan menemukan jalur terbaik melalui grafik. Algoritma ini terinspirasi oleh perilaku semut dalam menemukan jalur dari koloninya menuju makanan.

Untuk mendiskusikan algoritma semut, lingkungan yang digunakan adalah sebuah graf yang *fully connected* (setiap *node* memiliki busur ke *node* yang lain) dan *bidirectional* (setiap jalur bisa ditempuh bolak-balik dua arah).). Setiap busur memiliki bobot yang menunjukkan jarak antara dua buah *nodes* yang dihubungkan oleh busur tersebut. Untuk selanjutnya, setiap semut akan mencari rute menuju makanan yang dituju hingga akhirnya koloni semut menemukan jalur optimal yang akan dilalui menuju makanan tersebut. Proses mencari jalan optimal tersebut tidak semudah yang kita bayangkan dan tentu saja memerlukan rumus yang pada intinya menerapkan suatu fungsi *heuristic* yang cukup rumit.

Ide algoritma koloni semut adalah untuk meniru perilaku ini melalui 'semut tiruan' berjalan seputar grafik yang menunjukkan persoalan yang harus diselesaikan. Ada banyak sekali penerapan algoritma semut dalam berbagai persoalan kehidupan sehari-hari. Persoalan-persoalan tersebut adalah *Traveling Salesman Problem* (TSP), *Quadratic Assignment Problem* (QAP), *Job-shop Scheduling Problem* (JSP), dan sejumlah aplikasi lain mencakup pengaturan jalur kendaraan, pewarnaan graf, dan network routing.

***Kata kunci**: Algoritma semut, Ant Colony Optimization, Traveling Salesman Problem, graf, node.*

1. Pendahuluan

Seiring berkembangnya pemikiran, banyak sekali ditemukan sejumlah algoritma dalam AI (*Artifical Intelligence*) yang mendapat inspirasi dari alam. Algoritma tersebut di antaranya adalah:

- cara kerja otak yang luar biasa mengilhami *neural network*,
- *genetic algorithm* belajar dari proses evolusi,
- dan dari semut kita menyeselaikan masalah optimisasi.

Optimisasi banyak dilakukan pada berbagai dalam kehidupan sehari-hari. Cara untuk melakukan optimisasi tersebut juga bermacam-macam, salah satunya adalah dengan menggunakan *Ant Colony Optimization* atau Algoritma Semut.

“Sesungguhnya dalam penciptaan langit dan bumi, dan silih bergantinya malam dan siang terdapat tanda-tanda bagi orang-orang yang berakal,” (QS 3:190)

2. Optimisasi

2.1 Definisi Optimisasi

Optimisasi ialah suatu proses untuk mencapai hasil yang ideal atau **optimal** (nilai efektif yang dapat dicapai).

Dalam disiplin matematika optimisasi merujuk pada studi permasalahan yang mencoba untuk mencari nilai minimal atau maksimal dari suatu fungsi nyata. Untuk dapat mencapai nilai *optimal* baik minimal atau maksimal tersebut, secara sistematis dilakukan pemilihan nilai variabel integer atau nyata yang akan memberikan solusi optimal.

2.2 Definisi Nilai Optimal

Nilai optimal adalah nilai yang didapat dengan melalui suatu proses dan dianggap menjadi suatu solusi jawaban yang paling baik dari semua solusi yang ada.

Nilai optimal dapat dicari dengan dua cara, yaitu:

- cara konvensional, yaitu mencoba semua kemungkinan yang ada dengan mencatat nilai yang didapat cara ini kurang efektif, karena optimasi akan berjalan secara lambat.
- cara kedua adalah dengan menggunakan suatu rumus atau gambar sehingga nilai optimal dapat diperkirakan dengan cepat dan tepat.

2.3 Macam-macam Persoalan Optimisasi

Persoalan yang berkaitan dengan optimisasi sangat kompleks dalam kehidupan sehari-hari. Nilai optimal yang didapat dalam optimisasi dapat berupa besaran panjang, waktu, jarak, dan lain-lain.

Berikut ini adalah beberapa persoalan yang memerlukan optimisasi:

1. Menentukan lintasan terpendek dari suatu tempat ke tempat yang lain.
2. Menentukan jumlah pekerja seminimal mungkin untuk melakukan suatu proses produksi agar pengeluaran biaya pekerja dapat diminimalkan dan hasil produksi tetap maksimal.
3. Mengatur jalur kendaraan umum agar semua lokasi dapat dijangkau.
4. Mengatur *routing* jaringan kabel telepon agar biaya pemasangan kabel tidak terlalu besar dan penggunaannya tidak boros.

Selain beberapa contoh di atas, masih banyak persoalan lainnya yang terdapat di kehidupan sehari-hari.

3 Algoritma Semut

3.1 Sejarah Algoritma Semut

Pada tahun 1996, dunia AI pun ikut belajar dari semut dengan diperkenalkannya algoritma semut, atau *Ant Colony Optimization*, sebagai sebuah simulasi multi agen yang menggunakan metafora alami semut untuk menyelesaikan *problem* ruang fisik.

Algoritma semut diperkenalkan oleh **Moyson** dan **Manderick** dan secara meluas dikembangkan oleh **Marco Dorigo**, merupakan teknik probabilistik untuk menyelesaikan masalah komputasi dengan menemukan jalur terbaik melalui grafik. Algoritma ini terinspirasi oleh perilaku semut dalam menemukan jalur dari koloninya menuju makanan.

3.2 Cara kerja semut mencari jalur optimal

Semut mampu mengindera lingkungannya yang kompleks untuk mencari makanan dan kemudian kembali ke sarangnya dengan meninggalkan zat feromon pada jalur-jalur yang mereka lalui.

Feromon adalah zat kimia yang berasal dari kelenjar endokrin dan digunakan oleh makhluk hidup untuk mengenali sesama jenis, individu lain, kelompok, dan untuk membantu proses reproduksi.Berbeda dengan hormon, feromon menyebar ke luar tubuh dan hanya dapat mempengaruhi dan dikenali oleh individu lain yang sejenis (satu spesies).

Proses peninggalan feromon ini dikenal sebagai *stigmergy*, sebuah proses memodifikasi lingkungan yang tidak hanya bertujuan untuk mengingat jalan pulang ke sarang, tetapi juga memungkinkan para semut berkomunikasi dengan koloninya.

Seiring waktu, bagaimanapun juga jejak feromon akan menguap dan akan mengurangi kekuatan daya tariknya. Lebih lama seekor semut pulang pergi melalui jalur tersebut, lebih lama jugalah feromon menguap.

Agar semut mendapatkan jalur optimal, diperlukan beberapa proses:

1. Pada awalnya, semut berkeliling secara acak, hingga menemukan makanan. Lihat Gambar 1.

2. Ketika menemukan makanan mereka kembali ke koloninya sambil memberikan tanda dengan jejak feromon.

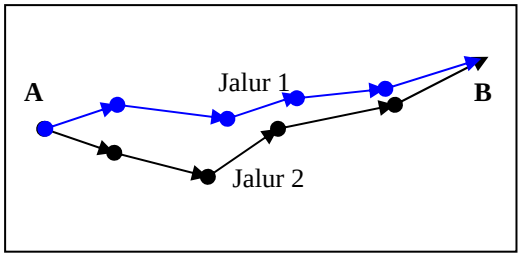
3. Jika semut-semut lain menemukan jalur tersebut, mereka tidak akan bepergian dengan acak lagi, melainkan akan mengikuti jejak tersebut.

4. Kembali dan menguatkannya jika pada akhirnya mereka pun menemukan makanan.

5. Seekor semut yang secara tidak sengaja menemukan jalur optimal akan menempuh jalur ini lebih cepat dari rekan-rekannya, melakukan *round-trip* lebih sering, dan dengan sendirinya meninggalkan feromon lebih banyak dari jalur-jalur yang lebih lambat ditempuh.

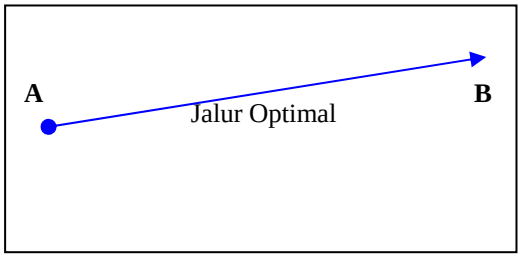
6. Feromon yang berkonsentrasi tinggi pada akhirnya akan menarik semut-semut lain untuk berpindah jalur, menuju jalur paling optimal, sedangkan jalur lainnya akan ditinggalkan.

7. Pada akhirnya semua semut yang tadinya menempuh jalur yang berbeda-beda akan beralih ke sebuah jalur tunggal yang ternyata paling optimal dari sarang menuju ke tempat makanan. Lihat Gambar 2.



Gambar 1. Lintasan Awal Semut Menuju Tempat Makanan

Keterangan Gambar 1:
A : Tempat awal koloni (sarang)
B : Tujuan koloni semut (makanan)
Jalur 1 (biru): Lintasan yang ditempuh oleh semut 1
Jalur 2 (hitam): Lintasan yang ditempuh oleh semut 2



Gambar 2. Lintasan Optimal Semut Menuju Tempat Makanan

Keterangan Gambar 2:
A : Tempat awal koloni (sarang)
B : Tujuan koloni semut (makanan)
Jalur Optimal : Jalur yang dilewati semut setelah beberapa iterasi

Seluruh proses ini menunjukkan berlangsungnya optimisasi alami kaum semut yang bisa kita tiru dalam kehidupan sehari-hari.

4 Graf

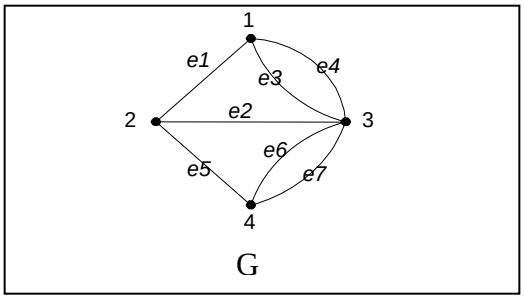
4.1 Definisi Graf

Graf digunakan untuk merepresentasikan objek-objek diskrit dan hubungan antara objek-objek tersebut.

Graf $G = (V, E)$, yang dalam hal ini:

V = himpunan tidak-kosong dari simpul-simpul (*vertices*)
= $\{ v_1 , v_2 , \ldots , v_n \}$

E = himpunan sisi (*edges*) yang menghubungkan sepasang simpul
= $\{e_1 , e_2 , ... , e_n \}$



Gambar 3. Contoh Graf

Keterangan Gambar 3:
G adalah graf dengan
V = { 1, 2, 3, 4 }
E = { (1, 2), (2, 3), (1, 3), (1, 3), (2, 4), (3, 4), (3, 4) }
= { e_1, e_2, e_3, e_4, e_5, e_6, e_7}

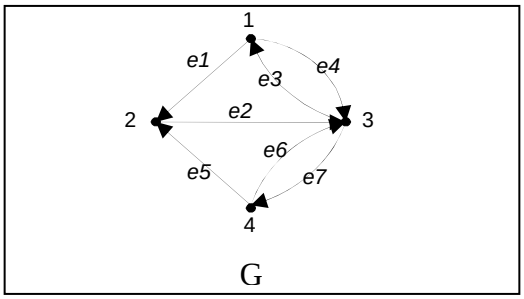
Berdasarkan orientasi arah pada sisi, maka secara umum graf dibedakan atas 2 jenis:

1. Graf tak-berarah (*undirected graph*)

Graf yang sisinya tidak mempunyai orientasi arah disebut graf tak-berarah.

2. Graf berarah (*directed graph* atau *digraph*)

Graf yang setiap sisinya diberikan orientasi arah disebut sebagai graf berarah.

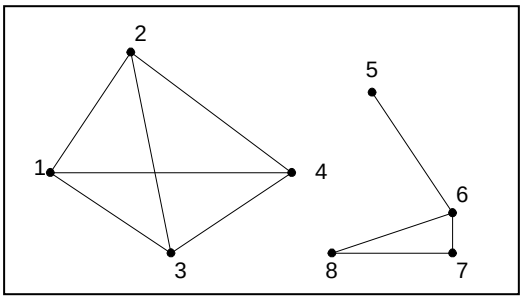


Gambar 4. Graf Berarah

Dua buah simpul v_1 dan simpul v_2 disebut **terhubung** jika terdapat lintasan dari v_1 ke v_2.

G disebut **graf terhubung** (*connected graph*) jika untuk setiap pasang simpul v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j.

Jika tidak, maka G disebut **graf tak-terhubung** (*disconnected graph*).



Gambar 5. Contoh graf tak-terhubung:

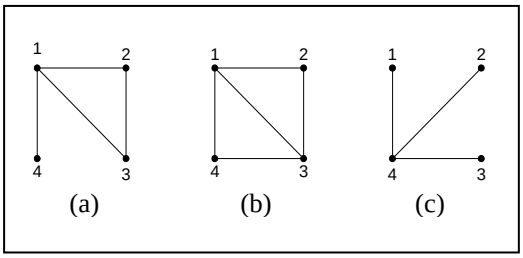
Graf berarah G dikatakan terhubung jika graf tidak berarahnya terhubung (graf tidak berarah dari G diperoleh dengan menghilangkan arahnya).

4.2 Graf Hamilton

Lintasan Hamilton ialah lintasan yang melalui tiap simpul di dalam graf tepat satu kali.

Sirkuit Hamilton ialah sirkuit yang melalui tiap simpul di dalam graf tepat satu kali, kecuali simpul asal (sekaligus simpul akhir) yang dilalui dua kali.

Graf yang memiliki sirkuit Hamilton dinamakan **graf Hamilton**, sedangkan graf yang hanya memiliki lintasan Hamilton disebut **graf semi-Hamilton**.



Gambar 6. Penggambaran Graf Hamilton

Keterangan Gambar 6:
(a) graf yang memiliki lintasan Hamilton (misal: 3, 2, 1, 4)
(b) graf yang memiliki lintasan Hamilton (1, 2, 3, 4, 1)
(c) graf yang tidak memiliki lintasan maupun sirkuit Hamilton

5. Analisis Algoritma Semut untuk Mencari Nilai Optimal Menggunakan Graf

Untuk mendiskusikan algoritma semut, lingkungan yang akan kita gunakan adalah sebuah graf yang *fully connected* (setiap *node* memiliki busur ke *node* yang lain) dan *bidirectional* (setiap jalur bisa ditempuh bolak-balik dua arah). Setiap busur memiliki bobot yang menunjukkan jarak antara dua buah *nodes* yang dihubungkan oleh busur tersebut.

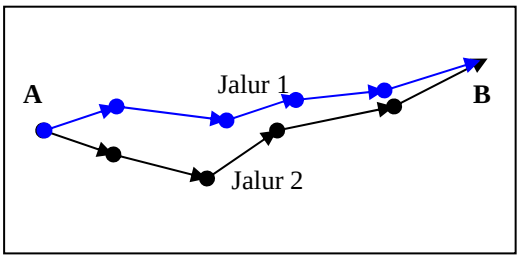
Algoritma ini menggunakan sistem multi agen, yang berarti kita akan mengerahkan seluruh koloni semut yang masing-masingnya bergerak sebagai agen tunggal. Setiap semut menyimpan daftar tabu yang memuat *nodes* yang sudah pernah ia lalui, dimana ia tidak diijinkan untuk melalui *node* yang sama dua kali dalam satu kali perjalanan (daftar ini disebut juga sebagai jalur Hamilton, yaitu jalur pada graf dimana setiap *node* hanya dikunjungi satu kali).

Sebuah koloni semut diciptakan, dan setiap semut ditempatkan pada masing-masing *node* secara merata untuk menjamin bahwa tiap *node* memiliki peluang untuk menjadi titik awal dari jalur optimal yang dicari. Setiap semut selanjutnya harus melakukan tur semut, yaitu perjalanan mengunjungi semua *nodes* pada graf tersebut.

Berikut adalah tahapan-tahapan algoritma semut menggunakan graf:

1. Dari sarang, semut berkeliling secara acak mencari makanan sambil mencatat jarak antara *node* yang ia lalui.

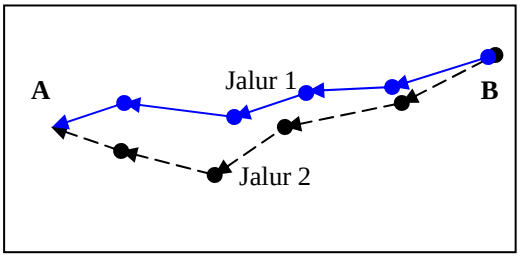
2. Ketika sampai ke makanan, Total jarak dari tiap *node* yang ia tempuh dijumlahkan untuk mendapatkan jarak dari sarang ke makanan.



Gambar 7. Lintasan Awal Semut Menuju Tempat Makanan

Keterangan Gambar 7:
A : Tempat awal koloni (sarang)
B : Tujuan koloni semut (makanan)
Jalur 1 (biru): Lintasan yang ditempuh oleh semut 1
Jalur 2 (hitam): Lintasan yang ditempuh oleh semut 2

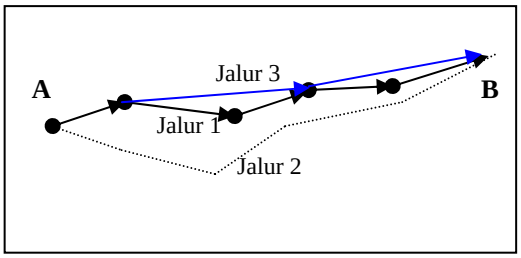
3. Ketika kembali ke sarang, sejumlah konsentrasi feromon ditambahkan pada jalur yang telah ditempuh berdasarkan total jarak jalur tersebut. Makin kecil total jarak (atau makin optimal), maka makin banyak kadar feromon yang dibubuhkan pada masing-masing busur pada jalur tersebut.



Gambar 8. Lintasan Semut Menuju Sarang

Keterangan Gambar 8:
A : Sarang semut
B : Tempat ditemukannya makanan
Jalur 1 (biru) : Jalur yang ditempuh oleh semut 1 dengan pemberian kadar feromon yang tinggi
Jalur 2 (hitam) : Jalur yang ditempuh oleh semut 2 dengan pemberian kadar feromon yang rendah

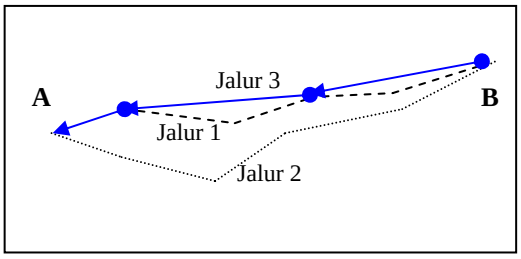
4. Untuk memilih busur mana yang harus dilalui berikutnya, digunakan sebuah rumus yang pada intinya menerapkan suatu fungsi *heuristic* untuk menghitung intensitas feromon yang ditinggalkan pada suatu busur.



Gambar 9. Lintasan Semut Menuju Makanan pada Iterasi ke-2

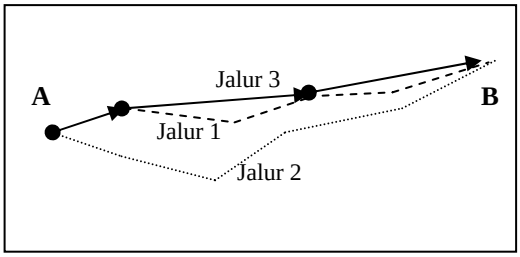
Keterangan Gambar 9:
A : Sarang semut
B : Tempat ditemukannya makanan
Jalur 1 : Jalur yang ditempuh oleh semut 1 karena kadar feromon yang tinggi
Jalur 2 : Jalur yang tidak ditempuh oleh semut karena kadar feromon yang rendah
Jalur 3 : Jalur yang ditemukan oleh semut 2

5. Pada iterasi berikutnya, busur-busur yang mengandung feromon lebih tinggi ini akan cenderung dipilih sebagai busur yang harus ditempuh berikutnya berdasarkan rumus pemilihan busur. Akibatnya, lama-kelamaan akan terlihat **jalur optimal** pada graf, yaitu jalur yang dibentuk oleh busur-busur dengan kadar feromon yang tinggi, yang pada akhirnya akan dipilih oleh semua multi agen semut.



Gambar 10. Lintasan Semut Menuju Sarang pada Iterasi ke-2

Keterangan Gambar 10:
A : Sarang semut
B : Tempat ditemukannya makanan
Jalur 1 (hitam) : Jalur yang ditempuh oleh semut 2 dengan pemberian kadar feromon yang rendah
Jalur 2 : Jalur yang tidak ditempuh
Jalur 3 (biru) : Jalur yang ditempuh oleh semut 2 dengan pemberian kadar feromon yang tinggi



Gambar 11. Lintasan Optimal Semut Menuju Tempat Makanan

Keterangan Gambar 11:
A : Sarang semut
B : Tempat ditemukannya makanan
Jalur 1 : : Jalur yang tidak ditempuh karena kadar feromon yang rendah
Jalur 2 : Jalur yang tidak ditempuh karena kadar feromon yang sangat rendah
Jalur 3 : Jalur optimal yang ditempuh oleh semut karena kadar feromon yang tinggi

6. Penerapan Algoritma Semut

Algoritma optimisasi koloni semut telah digunakan untuk menghasilkan penyelesaian yang mendekati optimal. Aplikasi algoritma semut dalam kehidupan sehari-hari mencakup beberapa persoalan, yaitu:

1. *Traveling Salesman Problem* (TSP), yaitu mencari jalur terpendek dalam sebuah graf menggunakan jalur Hamilton.

2. *Quadratic Assignment Problem* (QAP) yang berusaha meng-assign sejumlah n resources untuk ditempatkan pada sejumlah m lokasi dengan meminimalisir biaya assignment.

3. *Job-shop Scheduling Problem* (JSP) juga salah satu contoh aplikasi algoritma semut untuk menjadwalkan sejumlah j pekerjaan menggunakan sejumlah m mesin demikian sehingga seluruh pekerjaan diselesaikan dalam waktu yang minimal.

4. pengaturan jalur kendaraan

5. pewarnaan graf

6. network routing, dll.

6.1 Traveling Salesman Problem (TSP)

6.1.1 Contoh Kasus

Travelling Salesman Problem (TSP) adalah suatu masalah yang ditemukan oleh pedagang yang harus bepergian dan singgah di beberapa kota hingga kembali ke kota semula.

Berikut adalah contoh kasus TSP:

"Diberikan sejumlah kota dan jarak antar kota. Tentukan sirkuit terpendek yang harus dilalui oleh seorang pedagang bila pedagang itu berangkat dari sebuah kota asal dan menyinggahi setiap kota tepat satu kali dan kembali lagi ke kota asal keberangkatan."

Apabila kita mengubah contoh kasus tersebut menjadi persoalan pada graf, maka dapat dilihat bahwa kasus tersebut adalah bagaimana menentukan sirkuit Hamilton yang memiliki bobot minimum pada graf tersebut.

Dalam kehidupan sehari-hari, kasus TSP ini dapat diaplikasikan untuk menyelesaikan kasus lain, yaitu:

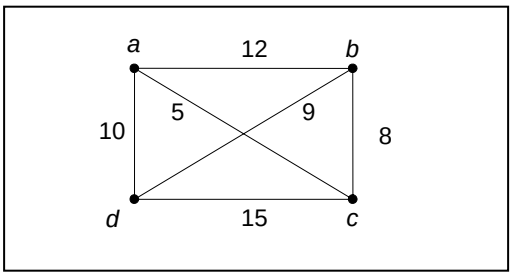
1. Pak Pos mengambil surat di kotak pos yang tersebar pada *n* buah lokasi di berbagai sudut kota.

2. Lengan robot mengencangkan *n* buah mur pada beberapa buah peralatan mesin dalam sebuah jalur perakitan.

3. Produksi *n* komoditi berbeda dalam sebuah siklus.

Seperti kita ketahui, bahwa untuk mencari jumlah sirkuit Hamilton di dalam graf lengkap dengan n simpul adalah: (*n* - 1)!/2.

Contoh:



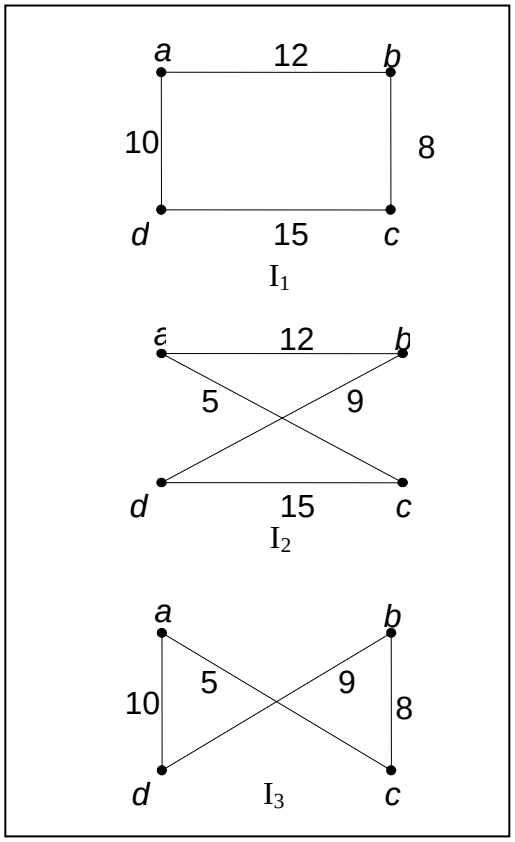
Gambar 12. Graf Lengkap

Graf di atas memiliki (4 – 1)!/2 = 3 sirkuit Hamilton (Gambar 12), yaitu:

I_1 = (*a*, *b*, *c*, *d*, *a*) atau (*a*, *d*, *c*, *b*, *a*) ==> panjang = 10 + 12 + 8 + 15 = 45

I_2 = (*a*, *c*, *d*, *b*, *a*) atau (*a*, *b*, *d*, *c*, *a*) ==> panjang = 12 + 5 + 9 + 15 = 41

I_3 = (*a*, *c*, *b*, *d*, a) atau (*a*, *d*, *b*, *c*, *a*) ==> panjang = 10 + 5 + 9 + 8 = 32



Gambar 13. Sirkuit Hamilton

Jadi, sirkuit Hamilton terpendek adalah I_3 = (*a*, *c*, *b*, *d*, *a*) atau (*a*, *d*, *b*, *c*, *a*) dengan panjang sirkuit = 10 + 5 + 9 + 8 = 32.

Jika jumlah simpul *n* = 20 akan terdapat (19!)/2 sirkuit Hamilton atau sekitar 6×10^{16} penyelesaian.

6.1.2 Penyelesaian TSP menggunakan algoritma semut

TSP adalah salah satu teka-teki optimisasi yang cukup terkenal di kalangan peneliti dan pecinta matematika selama bertahun-tahun. Mereka berlomba untuk mencari penyelesaian kasus TSP dengan tekniknya masing-masing. Teknik yang cukup terkenal adalah *simulated annealing*, *genetic algorithm*, and *ant colony optimization* (algoritma semut).

Dalam makalah ini, kita akan membahas teknik yang terakhir, yaitu algoritma semut.

Algoritma semut atau *Ant Colony Optimization* telah digunakan untuk mencari lintasan optimal pada *Travelling Salesman Problem* (TSP).

Pada simulasi algoritma semut, diperlukan tiga tabel besar (dengan dimensi n x n dimana n adalah banyaknya kota) untuk mencari lintasan optimal.

Tabel pertama adalah **tabel jarak (*distance array*)**, untuk menghitung seluruh jarak dari kota yang satu ke kota lainnya.

Tabel kedua adalah **tabel feromon (*pheromone array*)**, untuk menyimpan kadar feromon pada jalur antara seluruh kota.

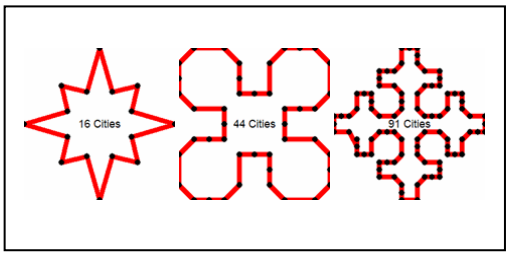
Tabel ketiga adalah **tabel delta feromon (*delta pheromone array*)**, untuk menyimpan sementara feromon untuk ditambahkan ke tabel feromon pada akhir iterasi. Tabel delta feromon digunakan agar semua semut mengetahui hasil dari iterasi sebelumnya.

Parameter kendali yang penting adalah jumlah semut, parameter untuk menghitung kemungkinan jalan selanjutnya, jumlah feromon untuk ditambahkan setiap kali perjalanan, dan nilai penguapan feromon.

Di bawah ini adalah *pseudo code* dari logika keseluruhan simulasi:

```
Create and locate a number of city (bot) objects on the screen
Create a tour (Group) object to hold the cities
Create a best tour (Group) object to hold the shortest tour found
Randomly shuffle the order of the cities in the tour
Create number of ants equal to number of cities
Initialize distance, pheromone, and delta pheromone edge arrays
For each iteration do
   For each ant in colony do:
      Do until all cities visited once:
         Decide which city to go to next:
            Calculate visibility (1/distance) to each city
            Sense pheromone trail to each city
            Choose move with prob related to (visibility^alpha) *
              (trail^beta)
         Move to next city
      Loop
      Calculate tour cost (distance)
      If cost of new tour is smaller than best tour then:
         Copy the tour to best tour
      End if
      Add pheromone to each edge of this ant's tour:
         Amount of pheromone to add based on 1/(tour cost)
         Add to delta pheromone array
   Next ant
   Evaporate pheromone array by some factor
   Add delta pheromone array to pheromone array
   Clear delta pheromone array
Next iteration
```

Berikut ini adalah contoh hasil penyelesaian TSP menggunakan algoritma semut. Lihat Gambar 14



Gambar 14. Penyelesaian TSP dengan 16, 44, dan 91 kota (*nodes*)

7. Kesimpulan

Algoritma semut merupakan suatu algoritma yang berasal dari alam. Algoritma ini terinspirasi oleh perilaku semut dalam menemukan jalur dari koloninya menuju makanan.

Algoritma ini menggunakan sistem multi agen, yang berarti kita akan mengerahkan seluruh koloni semut yang masing-masingnya bergerak sebagai agen tunggal. Setiap semut menyimpan daftar tabu yang memuat *nodes* yang sudah pernah ia lalui, dimana ia tidak diijinkan untuk melalui *node* yang sama dua kali dalam satu kali perjalanan (daftar ini disebut juga sebagai jalur Hamilton, yaitu jalur pada graf dimana setiap *node* hanya dikunjungi satu kali).

Untuk mendiskusikan penggunaan graf pada algoritma semut, lingkungan yang akan kita gunakan adalah sebuah graf yang *fully connected* (setiap *node* memiliki busur ke *node* yang lain) dan *bidirectional* (setiap jalur bisa ditempuh bolakbalik dua arah). Setiap busur memiliki bobot yang menunjukkan jarak antara dua buah *nodes* yang dihubungkan oleh busur tersebut.

Dalam proses perjalanan semut menuju makanan, terdapat suatu mekanisme untuk mencari lintasan optimal yang akan dilalui semut. Berikut prosesnya:

1. Dari sarang, semut berkeliling secara acak mencari makanan sambil mencatat jarak antara node yang ia lalui.

2. Ketika sampai ke makanan, Total jarak dari tiap node yang ia tempuh dijumlahkan untuk mendapatkan jarak dari sarang ke makanan.

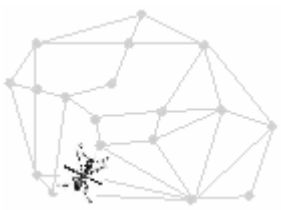
3. Ketika kembali ke sarang, sejumlah konsentrasi feromon ditambahkan pada jalur yang telah ditempuh berdasarkan total jarak jalur tersebut. Makin kecil total jarak (atau makin optimal), maka makin banyak kadar feromon yang dibubuhkan pada masing-masing busur pada jalur tersebut.

4. Untuk memilih busur mana yang harus dilalui berikutnya, digunakan sebuah rumus yang pada intinya menerapkan suatu fungsi heuristic untuk menghitung intensitas feromon yang ditinggalkan pada suatu busur.

5. Pada iterasi berikutnya, busur-busur yang mengandung feromon lebih tinggi ini akan cenderung dipilih sebagai busur yang harus ditempuh berikutnya berdasarkan rumus pemilihan busur. Akibatnya, lama-kelamaan akan terlihat jalur optimal pada graf, yaitu jalur yang dibentuk oleh busur-busur dengan kadar feromon yang tinggi, yang pada akhirnya akan dipilih oleh semua multi agen semut.

Algoritma optimisasi koloni semut telah digunakan untuk menghasilkan penyelesaian yang mendekati optimal. Aplikasi algoritma semut dalam kehidupan sehari-hari mencakup banyak persoalan, salah satunya adalah *Travelling Salesman Problem* (TSP).

Travelling Salesman Problem (TSP) adalah suatu masalah yang ditemukan oleh pedagang yang harus bepergian dan singgah di beberapa kota hingga kembali ke kota semula.



DAFTAR PUSTAKA

[1] Chietz. (2006). "Algoritma Semut". http://chietz.wordpress.com/2006/12/05/algoritma-semut/. Tanggal akses: 19 Desember 2006 pukul 21.30.

[2] Jamilah, Euis Widiani. (2005). "Perancangan dan Implementasi Algorima Semut untuk Menyelesaikan Masalah Spanning Tree." http://digilib.unikom.ac.id/. Tanggal akses: 20 Desember 2006 pukul 20.00.

[3] Marco, Dorigo. 1996. *Ant Colony System : A Cooperative Learning Approach to the Travelling Problem*, Université Libre de Bruxelles.

[4] Mike. (2006). "Visual Bots – Solving The TSP." http://www.visualbots.com/tsp_project.htm. Tanggal akses: 2 Januari 2007 pukul 09.30.

[5] Munir, Rinaldi. (2006). *Bahan Kuliah IF2151 Graf-1*. Departemen Teknik Informatika, Institut Teknologi Bandung.

[6] Munir, Rinaldi. (2006). *Bahan Kuliah IF2151 Graf-2*. Departemen Teknik Informatika, Institut Teknologi Bandung.

[7] Wikipedia Indonesia. (2006). "Algoritma Semut – Wikipedia Indonesia, ensiklopedia bebas berbahasa Indonesia." http://id.wikipedia.org/wiki/Algoritma_semut/. Tanggal akses: 19 Desember 2006 pukul 21:00.

[8] Wikipedia Indonesia. (2006). "Ant colony optimization - Wikipedia, the free encyclopedia." http://en.wikipedia.org/wiki/Ant_colony_optimization/. Tanggal akses: 19 Desember 2006 pukul 21:30.

[9] Wikipedia Indonesia. (2006). "Feromon – Wikipedia Indonesia, ensiklopedia bebas berbahasa Indonesia." http://id.wikipedia.org/wiki/Feromon/. Tanggal akses: 3 Januari 2007 pukul 15:30.

[10] Wikipedia Indonesia. (2006). "Optimisasi – Wikipedia Indonesia, ensiklopedia bebas berbahasa Indonesia." http://id.wikipedia.org/wiki/Optimisasi/. Tanggal akses: 2 Januari 2007 pukul 09:00.