Ant Colony System untuk Penyelesaian Masalah *Travelling Salesman Problem*

Aris Feryanto

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung
Jl. Ganesha 10 Bandung
e-mail: aris_feryanto@yahoo.com

ABSTRAK

Banyak algoritma telah diajukan untuk menyelesaikan persoalan *travelling salesman problem* (TSP) secara mangkus. Salah satunya adalah algoritma *ant colony optimization* (ACO) yang merupakan salah satu metode yang sukses menggunakan pendekatan yang berasal dari alam untuk mendesain sebuah algoritma optimasi. Makalah ini membahas *ant colony system* (ACS), sebuah algoritma dalam kelompok ACO yang diaplikasikan untuk masalah TSP. ACS mensimulasi-kan sekumpulan semut yang bekerja sama untuk mencari solusi yang baik untuk TSP. Simulasi ini meniru aktivitas semut pada dunia nyata yang telah diteliti dapat menentukan jalur terpendek dari sumber makanan ke sarangnya. Semut-semut bekerja sama dengan berkomunikasi melalui media hormon feromon yang dilepaskan sepanjang sisi pada graf TSP pada saat membentuk solusi. Beberapa hasil percobaan disertakan untuk menunjukkan performa ACS yang baik dalam menangani masalah TSP.

Kata kunci: *Ant colony optimization*, *Travelling salesman problem*, *Ant colony system*, Kata kunci 5.

1. PENDAHULUAN

Permasalahan *travelling salesman problem* atau sering disingkat TSP adalah masalah pencarian sebuah siklus tur yang mengunjungi semua kota tepat satu kali dalam himpunan kota yang diberikan dan kembali ke kota asal. Hingga saat ini belum ditemukan algoritma yang mangkus untuk menyelesaikannya. Algoritma yang paling bagus sekalipun mempunyai kompleksitas waktu eksponensial pada kasus terburuk. Hingga saat ini, banyak peneliti telah mencoba mencari pendekatan untuk menyelesaikan permasalahan TSP dalam waktu yang lebih singkat. Namun, karena hanya berupa pendekatan, hasilnya tidak dijamin pasti optimal, tapi rata-rata mendekati optimal.

Beberapa pendekatan yang telah diketahui hingga saat ini, antara lain *simulated annealing* (SA), *neural nets* (NNs), seperti *elastic net* (EN) dan *self organizing map* (SOM), *evolutionary computation* (EC), seperti *genetic algorithm* (GA) dan *evolutionary programming* (EP), kombinasi *simulated annealing* dan *genetic algorithm* (AG), serta *ant colony optimization* (ACO).

Penelitian mengenai algoritma ACO, terutama dalam aplikasinya untuk permasalahan TSP, telah dilakukan oleh Marco Dorigo (Belgia) dan Luca Maria Gambardella (Switzerland) pada tahun 1996. Berdasarkan *paper* [1] yang ditulis oleh mereka, *ant colony system* (ACS) yang merupakan salah satu algoritma ACO memiliki performa yang jauh lebih baik dari algoritma lain. Salah satu data di *paper* tersebut menunjuk-kan pada kasus TSP dengan 75 kota, ACS hanya membutuhkan simulasi tur sebanyak 3.480 kali untuk menemukan jalur tur terbaik, sedangkan *genetic algorithm* membutuhkan 80.000 kali simulasi tur untuk menemukan jalur tur terbaik dan algoritma lain seperti EP, SA dan AG bahkan membutuhkan jumlah simulasi tur yang jauh lebih banyak lagi.

Selanjutnya, akan dibahas bagaimana algoritma *ant colony optimization* dapat diterapkan untuk menyelesaikan persoalan TSP dimana untuk setiap kota memiliki jalur ke semua kota lainnya (*strongly connected graph*).

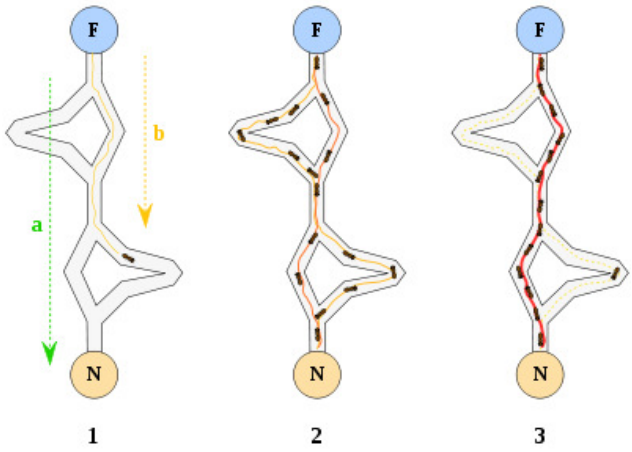
2. ANT COLONY OPTIMIZATION

2.1 Deskripsi

Apabila kita mengamati semut dalam dunia nyata, semut bergerak secara acak mencari makanan untuk dibawa pulang ke koloninya. Ketika pulang ke koloni dengan membawa makanan, semut menandai jalur pulangnya dengan hormon feromon [3]. 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 feromon lagi.

Feromon dapat menguap dengan cepat. Ketika feromon menguap, daya atraktifnya akan berkurang. Semakin lama waktu yang dibutuhkan oleh seekor semut untuk mengikuti jalur dan kembali lagi, semakin banyak feromon yang menguap. Sebaliknya, sebuah jalur yang pendek, akan dilewati lebih cepat dan kadar feromon tetap

tinggi karena feromon terus menerus diletakkan oleh semut yang lewat. Penguapan feromon ini juga berguna untuk menghindari kecenderungan tidak ditemukannya solusi optimal. Jika tidak ada penguapan sama sekali, jalur yang dipilih oleh semut yang pertama akan cenderung menjadi sangat atraktif untuk semut-semut lain, sehingga eksplorasi ruang solusi akan menjadi terbatas.



Gambar 1. Tingkah laku semut dalam dunia nyata

Ide dasar ACO berasal dari pengamatan bahwa semut dapat menemukan jalur terpendek dari sumber makanan ke sarangnya tanpa menggunakan petunjuk visual, hanya dengan feromon.

1. Semut pertama menemukan sumber makanan (F) dan pulang melalui jalur (a) kembali ke sarang (N). Semut kedua mengikuti jalur tersebut dan menemukan sumber makanan dan kembali ke sarang, namun melalui jalur (b) yang agak berbeda karena kadar feromon masih sangat sedikit.
2. Dengan asumsi semut-semut lainnya tidak membuat jalur baru, maka ada 4 kemungkinan jalan yang dapat dilalui. Namun jalur yang lebih pendek akan memiliki kadar feromon lebih tinggi karena selalu diperbarui oleh semut yang lewat sebelum feromon yang lama habis menguap.
3. Semut-semut akan mengambil jalur yang lebih pendek karena kadar feromon yang lebih tinggi dan jalur yang panjang akan kehilangan jalur feromonnya.

Dalam beberapa kali percobaan pada sebuah koloni semut dengan pilihan antara dua jalur yang tidak sama panjang menuju sumber makanan, ahli biologi mengamati bahwa semut-semut cenderung menggunakan rute yang paling pendek [3]. Model yang menjelaskan tingkah laku ini adalah sebagai berikut:

1. Seekor semut berjalan secara acak di sekitar koloni;
2. Jika semut itu menemukan sumber makanan, dia akan kembali ke sarang dan meninggalkan feromon sepanjang jalur yang dilaluinya;
3. Feromon ini bersifat atraktif, semut-semut yang ada di dekatnya akan cenderung mengikuti jalur ini;
4. Ketika kembali ke koloni, semut-semut ini akan memperkuat jalur;
5. Jika ada dua jalur yang mungkin untuk mencapai sumber makanan yang sama, jalur yang lebih pendek akan dilewati oleh lebih banyak semut daripada jalur yang panjang pada satu perioda waktu tertentu;
6. Jalur yang lebih pendek akan semakin diperkuat dengan feromon, oleh karena itu menjadi lebih atraktif;
7. Jalur yang panjang akan semakin menghilang karena feromon terus menguap;
8. Akhirnya, semua semut akan memilih jalur yang paling pendek.

Semut-semut menggunakan lingkungan sebagai media komunikasi. Mereka berkomunikasi secara tidak langsung dengan menjatuhkan feromon. Informasi ini hanya dipertukarkan pada lingkup lokal, hanya seekor semut yang berada di tempat feromon dijatuhkan yang menyadari keberadaaan feromon ini. Sistem ini disebut *stigmergy* dan terjadi di banyak lingkungan hewan.

2.2 Aplikasi

Algoritma *ant colony optimization* telah diaplikasikan pada banyak masalah optimasi kombinatorial, sesuai dengan judul penelitian yang melahirkan algoritma ini, yaitu penelitian "*Cooperation and learning for combinatorial optimization*" oleh Marco Dorigo dan Luca Gambardella. Algoritma ini juga digunakan untuk menghasilkan solusi yang hampir optimal untuk permasalahan TSP. Algoritma ini mempunyai keunggulan dibandingkan pendekatan *simulated annealing* dan *genetic algorithm* karena dapat beradaptasi dengan graf yang berubah secara dinamis.

3. APLIKASI PADA TSP

Pada penyelesaian masalah TSP ini, digunakan semut-semut buatan yang memiliki sifat yang hampir sama dengan semut pada kehidupan nyata. Sebuah semut buatan bergerak dari kota ke kota dalam sebuah graf TSP. Semut ini memilih kota yang akan dikunjungi berdasarkan fungsi probabilistik baik dari feromon yang terakumulasi pada sisi graf dan sebuah nilai heuristik, yang merupakan sebuah fungsi dari panjang sisi graf. Pada setiap langkah, semut-semut bergerak ke kota baru dan memodifikasi jalur feromon pada sisi yang dilewati – *local trail updating*. Ketika semua semut telah menyelesaikan sebuah tur, semut dengan tur paling pendek memodifikasi sisi-sisi yang termasuk dalam turnya – *global trail updating* – dengan menambahkan sejumlah feromon yang berbanding terbalik dengan panjang turnya.

Berikut ini adalah tiga ide dasar yang diambil dari tingkah laku semut alami yang ditransfer ke dalam koloni semut buatan:

1. Ketertarikan pada jalur dengan tingkat feromon yang lebih tinggi;
2. Pertambahan jumlah feromon yang lebih cepat pada jalur yang lebih pendek;
3. Jalur feromon yang menjadi media komunikasi diantara semut-semut.

Semut buatan dalam aplikasi penyelesaian masalah TSP ini juga diberi beberapa kemampuan tambahan yang tidak ada pada semut alami, yaitu kemampuan untuk mengingat kota mana saja yang sudah dikunjunginya dan kemampuan untuk menentukan seberapa jauh jarak dari satu kota ke kota lainnya.

3.1 *Ant Colony System*

Ada beberapa cara untuk memodelkan tiga ide dasar di atas menjadi model komputasi untuk menyelesaikan permasalahan TSP. Salah satunya, seperti yang diajukan oleh Dorigo dan Gambardella [1], adalah *ant colony system* (ACS). Dalam ACS, sebuah semut buatan k yang berada di kota r, memilih bergerak menuju kota s yang belum pernah dikunjungi sebelumnya (tidak tercatat dalam memori M_k, memori semut k) dengan menggunakan formula probabilitas sebagai berikut:

$$s = \begin{cases} \arg\max_{u \notin M_k} \{[\tau(r,s)] \cdot [\eta(r,s)]^{\beta}\}, & \text{if } q \le q_0 \\ S, & \text{sebaliknya} \end{cases}$$

dimana $\tau(r,s)$ adalah jumlah jalur feromon pada sisi (r,s), $\eta(r,s)$ adalah fungsi heuristik, yaitu invers dari jarak antara kota r dan kota u, β adalah sebuah parameter yang menjadi bobot relatif antara jalur feromon dan kedekatan kota, q adalah sebuah nilai yang dipilih secara acak dengan probabilitas seragam antara 0 sampai 1 (inklusif), q_0 $(0 \le q_0 \le 1)$ adalah sebuah parameter, dan S adalah sebuah variabel acak yang dipilih berdasarkan distribusi probabilitas berikut ini, yang lebih memilih jalan yang lebih pendek dan memiliki feromon yang lebih tinggi:

$$p_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)] \cdot [\eta(r,s)]^{\beta}}{\sum_{u \notin M_k} [\tau(r,s)] \cdot [\eta(r,s)]^{\beta}}, & \text{if } s \notin M_k \\ 0, & \text{sebaliknya} \end{cases}$$

dimana $p_k(r,s)$ adalah kemungkinan seekor semut k memilih untuk pergi dari kota r ke kota s.

Dalam ACS, jalur feromon diperbarui layaknya sifat feromon yang menguap pada dunia nyata. Pembaruan ini dilakukan secara lokal dan global. Pembaruan global bertujuan untuk meningkatkan peluang untuk melalui jalan yang lebih pendek. Ketika semut-semut buatan telah menyelesaikan turnya, semut dengan tur paling pendek meletakkan feromon pada sisi-sisi yang dikunjunginya dalam turnya. Jumlah feromon $\Delta\varphi(r,s)$ yang diletakkan pada setiap sisi adalah berbanding tebalik dengan panjang tur. Peletakan feromon ini bertujuan untuk mensimulasikan sifat dari perubahan akumulasi feromon, dimana pada kasus semut nyata disebabkan oleh panjang jalur dan berjalannya waktu. Formula untuk *global trail updating* adalah:

$$\varphi(r,s) \leftarrow (1-\alpha) \cdot \varphi(r,s) + \alpha \cdot \Delta\varphi(r,s)$$

dimana $\Delta\varphi(r,s) = (\text{tur terpendek})^{-1}$. *Global trail updating* adalah sebuah skema pembelajaran dimana solusi yang lebih baik mendapatkan penekanan yang lebih besar.

Pembaruan lokal ditujukan untuk menghindari sebuah sisi yang sangat kuat dipilih oleh semua semut. Pada setiap kali sebuah sisi dipilih oleh seekor semut, jumlah feromon diubah dengan menggunakan formula *local trail updating*:

$$\tau(r,s) \leftarrow (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \tau_0$$

dimana τ_0 adalah sebuah parameter. *Local trail updating* juga merupakan representasi penguapan feromon pada semut asli.

4. PERCOBAAN

Untuk menguji kemangkusan algoritma ACO, khususnya dengan sistem ACS, saya menggunakan program ACO untuk TSP yang diunduh dari http://iridia.ulb.ac.be/~mdorigo/ACO/downloads/ACOTSA.V1.0.tar.gz. Program ini ditulis dalam bahasa C dan memiliki beberapa implementasi algoritma ACO untuk persoalan TSP yang simetrik, diantaranya *ant system*, *elitist ant system*, *MAX-MIN ant system*, *rank-based version of ant system*, *best-worst ant system*, dan *ant colony system* (ACS).

Karena yang dibahas dalam makalah ini adalah algoritma ACS, program dijalankan dengan parameter "`--acs`" untuk menjalankan algoritma ACS. Program dijalankan dalam lingkungan Linux dengan virtualisasi dengan *software VirtualBox* 3.0.6 (dijalankan pada host Windows 7 RC), dengan prosesor *dual-core* 2.2 GHz.

Masukan program diambil dari pranala http://elib.zib.de/pub/mp-testdata/tsp/tsplib/, yaitu untuk masalah TSP simetrik dengan nama file *pcb442.tsp*. Isi file ini berupa koordinat kartesian 442 kota yang akan dicari solusi TSP-nya. Adapun solusi optimal untuk permasalahan 442 kota ini telah ditemukan dan dapat dilihat pada file *pcb442.opt.tour* pada pranala yang sama untuk file masukan, yaitu dengan panjang lintasan tur 50.778 satuan panjang. Pada hasil di bawah, kita akan membandingkan hasil algoritma ACS dengan solusi optimal yang telah ditemukan.

Parameter lengkap *command-line* untuk menjalankan program:

```
./acotsp --acs -i pcb442.tsp
```

4.1 Hasil

Berikut ini adalah hasil beberapa percobaan menggunakan program. Setiap solusi yang didapatkan pada tabel berikut merupakan hasil terbaik (*best solution*) dari eksekusi tiap percobaan selama maksimum 10 detik.

Tabel 1 Hasil percobaan algoritma ACS dengan program

Percobaan ke	Rute Terpendek	Ditemukan pada iterasi ke	Ditemukan pada waktu (detik)
1	50792	207	4.98
2	50931	211	5.00
3	50931	226	5.36
4	50922	260	8.12
5	50931	85	2.08
6	50916	203	7.03
7	50778	376	9.04
8	50778	293	8.77
9	50927	157	5.28
10	50929	332	7.76

Rata-rata rute terpendek: 50883,50
Rata-rata iterasi untuk mendapatkan rute terpendek: 235
Rute terpendek yang terbaik: 50778
Rute terpendek yang terburuk: 50931

Dapat dilihat dari hasil percobaan, dalam sepuluh kali percobaan, didapatkan dua hasil panjang rute terpendek yang sama dengan panjang rute optimal, yaitu 50778. Ini membuktikan bahwa algoritma ACS mampu mendekati solusi optimum, bahkan terkadang menghasilkan solusi yang optimum.

4.2 Perbandingan

Apabila permasalahan TSP di atas diselesaikan dengan algoritma *brute force*, maka akan dibutuhkan pengecekan untuk (442 – 1)! / 2 = $1{,}241 \times 10^{976}$ kemungkinan. Jika diasumsikan satu pengecekan kemungkinan memakan waktu 1 *clock* CPU (pada kenyataannya pasti memakan lebih dari 1 *clock*), maka pada prosesor *dual-core* 2.2 GHz, pengecekan semua kemungkinan akan memakan waktu $2{,}821 \times 10^{966}$ detik atau $3{,}265 \times 10^{961}$ hari atau $8{,}947 \times 10^{958}$ tahun. Sungguh suatu hal yang tidak mungkin dan sangat jauh dibandingkan dengan algoritma ACS yang hanya memakan waktu kurang dari 10 detik untuk menemukan solusi yang mendekati optimal.

Perbandingan dengan pendekatan lain seperti *genetic algorithm*, *evolutionary programming*, dan *simulated annealing* terdapat pada [1] dan menunjukkan bahwa ACS adalah algoritma yang menghasilkan hasil yang lebih mendekati optimal dalam waktu yang lebih cepat dari pendekatan-pendekatan lain.

4. KESIMPULAN

Algoritma *ant colony optimization* (ACO) merupakan pendekatan yang bagus untuk mencari solusi yang mendekati optimal, bahkan kadang-kadang mendapatkan solusi optimal untuk permasalahan *travelling salesman problem* (TSP). Dibandingkan dengan pendekatan-pendekatan lainnya, ACO memiliki keunggulan waktu pemrosesan yang sangat cepat dan kemampuan adaptasi dengan graf yang berubah secara dinamis. Dengan algoritma ini, sebagian besar masalah TSP pada dunia nyata, terutama yang menyangkut proses bisnis, dapat diselesaikan dengan hasil mendekati optimal dengan cepat. Algoritma ini dapat dikembangkan lagi untuk skala yang lebih besar dengan memanfaatkan fitur *parallel processing* yang terdapat pada prosesor-prosesor baru saat ini.

REFERENSI

[1] Gambardella, L. M. and Dorigo, M., "Ant colonies for the traveling salesman problem", 1997, http://www.idsia.ch/~luca/acs-bio97.pdf, Tanggal akses: 1 Januari 2009

[2] Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika ITB, 2009.

[3] Ant colony optimization – Wikipedia, http://en.wikipedia.org/wiki/Ant_colony_optimization, Tanggal akses: 1 Januari 2009

[4] Maria, Anna et al, "Penyelesaian Masalah Travelling Salesman Problem Menggunakan Ant Colony Optimization (ACO)", http://www.informatika.org/~rinaldi/Stmik/Makalah/MakalahStmik29.pdf, Tanggal akses: 1 Januari 2009

[5] ACO program for TSP, http://iridia.ulb.ac.be/~mdorigo/ACO/downloads/ACOTSP.V1.0.tar.gz, Tanggal akses: 1 Januari 2009

[6] TSP Solver for Google Maps, http://gebweb.net/optimap/, Tanggal akses: 1 Januari 2009

[7] Swarm intelligence – Wikipedia, http://en.wikipedia.org/wiki/Swarm_intelligence, Tanggal akses: 1 Januari 2009

[8] Pheromone – Wikipedia, http://en.wikipedia.org/wiki/Pheromone, Tanggal akses: 1 Januari 2009