Optimasi *Branch and Bound* pada Persoalan *Travelling Salesman Problem*

Hasna Nur Karimah - 13514106
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
hasnank@s.itb.ac.id

***Abstrak*—*Travelling Salesman Problem* (TSP) merupakan persoalan yang sampai sekarang masih dicari algoritmanya yang paling mangkus, terutama untuk kasus yang tidak kecil. Sudah banyak algoritma yang ditemukan, tetapi pengoptimalan masih terus dilakukan. Dalam makalah ini, akan dibahas beberapa pengoptimalan pada algoritma *Branch and Bound* yang diterapkan pada TSP.**

***Kata kunci*—ACO, *Branch and Bound*, Christofides, *cutting-plane method*, TSP.**

I. PENDAHULUAN

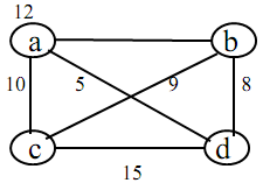
Travelling salesman problem (TSP) adalah permasalahan yang digambarkan dengan pertanyaan berikut: "Diketahui daftar kota dan jarak antara setiap dua kota, bagaimanakah jalur paling pendek yang memungkinkan untuk mengunjungi setiap kota tepat sekali dan kembali ke kota asal?"

Persoalan ini termasuk ke dalam persoalan NP-hard. Terdapat juga persoalan menentukan apakah suatu graf memiliki jalur yang lebih pendek dari suatu jarak *L*, persoalan ini disebut *travelling salesman decision problem* (TSDP). Persoalan ini termasuk ke dalam persoalan NP-complete.

Persoalan ini telah dipelajari sejak tahun 1930 dan sampai sekarang masih menjadi persoalan yang terus dipelajari dan dicari optimasinya. Walaupun sulit, sudah banyak cara yang ditemukan untuk menyelesaikan persoalan ini, baik dengan metode heuristic ataupun algoritma yang sudah pasti.

TSP dapat digambarkan sebagai permasalahan graf. Setiap kota dilambangkan sebagai simpul, sedangkan sisinya adalah jarak antar kota. Biasanya graf yang digunakan adalah graf tak berarah dengan tiap simpul terhubung ke simpul lainnya melalui satu sisi. Persoalan ini menjadi persoalan minimasi dengan sebuah titik sebagai titik awal dan akhir, dan setiap simpul lain dikunjungi tepat sekali.

TSP yang digambarkan dengan graf tak berarah disebut juga TSP simetri. Namun pada kenyataannya, jarak yang ditempuh seringkali tidak sama antara A ke B dan B ke A. Bukan hanya jarak, tetapi mungkin juga yang berbeda adalah waktu ataupun biaya yang dibutuhkan. Karena kemungkinan ketidaksamaan ini, ada kasus TSP yang digambarkan dengan graf berarah, disebut juga TSP asimetri. Ketidaksimetrian ini dapat disebabkan oleh factor-faktor seperti kemacetan, jalan searah, ongkos bolak-balik yang berbeda, dll.



Gambar 1 Persoalan TSP simetri dalam representasi graf tak-berarah (sumber: slide kuliah IF2211 Strategi Algoritma: Algoritma-Branch-&-Bound-(2016))

TSP asimetri ini dapat diselesaikan dengan terlebih dahulu mengubahnya menjadi TSP simetri. Misalnya diberikan sebuah matriks yang merepresentasikan jarak antar kota seperti di bawah ini:

Tabel 1 Matriks jarak asimetri

	A	*B*	*C*
A		1	2
B	6		3
C	5	4	

maka akan dibentuk sebuah matriks yang berukuran dua kalinya, dengan membentuk "simpul bayangan". Matriks baru yang simetri dari matriks di atas akan berbentuk seperti ini:

Tabel 2 Matriks jarak simetri

	A	*B*	*C*	*A'*	*B'*	*C'*
A				−∞	6	5
B				1	−∞	4
C				2	3	−∞
A'	−∞	1	2			
B'	6	−∞	3			
C'	5	4	−∞			

Selain TSP asimetri, ada juga kasus lain dalam TSP, yaitu metrik TSP. Metrik TSP, atau dikenal juga dengan sebutan delta-TSP, adalah TSP dengan jarak antarkota memenuhi ketidaksamaan segitiga. Jarak yang menghubungkan A dan B secara langsung tidak akan lebih besar dari jarak A dan B melalui C, atau dituliskan dalam persamaan:

$$d_{AB} \leq d_{AC} + d_{CB} \quad (1)$$

Banyak aplikasi dari TSP yang membolehkan suatu kota dikunjungi lebih dari sekali. Dalam aplikasi-aplikasi tersebut, TSP yang simetri dan non-metrik dapat direduksi ke TSP metrik. Graf semula akan digantikan oleh graf komplit, di mana jarak antarkota A dan B akan digantikan dengan jalur terpendek antara A dan B di graf semula.

TSP metrik ini juga memiliki kasus khusus lagi, yaitu TSP Euclidean atau disebut juga TSP planar, yaitu TSP dengan jarak Euclidean. Jarak pada bidang planar memenuhi ketidaksamaan segitiga, sehingga termasuk ke dalam TSP metrik. TSP Euclidean termasuk ke dalam persoalan NP-hard, tetapi jika metrik didiskritkan (jarak dibulatkan ke dalam bilangan bulat), persoalan menjadi NP-complete. Persoalan menjadi lebih mudah karena akan terbentuk pohon merentang minimum Euclidean di mana kompleksitas komputasinya adalah O (n log n) untuk n buah titik.

II. LANDASAN TEORI

Algoritma *Branch and Bound* (B&B) adalah algoritma dengan paradigm diskrit dan optimasi persoalan kombinatorial. Algoritma ini merupakan metode pencarian di dalam ruang solusi secara sistematis. Pembentukan pohon ruang status pada algoritma ini dibangun secara dinamis dengan skema BFS. Namun bedanya, simpul yang diekspansi bukanlah berdasarkan urutan pembangkitan, tetapi simpul yang memiliki "ongkos" paling kecil di antara simpul-simpul hidup lainnya.

Setiap simpul memiliki nilai "ongkos", yaitu nilai taksiran lintasan termurah dari simpul status i ke status tujuan, dan dapat dinyatakan secara umum sebagai:

$$\hat{c}(i) = \hat{f}(i) + \hat{g}(i) \quad (2)$$

di mana,

$\hat{c}(i)$ = ongkos untuk simpul i
$\hat{f}(i)$ = ongkos mencapai simpul i dari akar
$\hat{g}(i)$ = ongkos mencapai simpul tujuan dari simpul i.

Cara kerja algoritma B&B berdasarkan pada dua prinsip berikut ini.

- Pembagian ruang status secara rekursif menjadi ruang-ruang yang lebih kecil dan meminimalkan "ongkos" pada ruang-ruang tersebut. Proses ini dinamakan *branching*.
- *Branching* akan senilai dengan enumerasi secara *brute-force*. Untuk meningkatkan performansi, digunakan *bound* untuk membatasi ruang status yang dibangkitkan, mengeliminasi kandidat solusi yang terbukti tidak mengandung solusi optimal.

Sesuai dengan namanya, algoritma ini memiliki *bound* atau fungsi pembatas. Fungsi ini akan memangkas jalur yang dianggap tidak lagi mengarah ke solusi. Pemangkasan dilakukan pada simpul yang memiliki salah satu dari ciri berikut[1]:

- Nilai simpul tidak lebih baik dari nilai terbaik sejauh ini
- Simpul tidak merepresentasikan solusi yang *feasible* karena ada batasan yang dilanggar
- Solusi yang *feasible* pada simpul tersebut hanya terdiri atas satu titik, maka tidak ada pilihan lain. Dalam kasus ini, yang harus dilakukan adalah membandingkan nilai fungsi obyektif dengan solusi terbaik saat ini, dan yang terbaiklah yang diambil

Algoritma B&B dapat dituliskan seperti ini[1]:

1. Masukkan simpul akar ke dalam antrian Q. Jika simpul akar adalah simpul solusi (goal node), maka solusi telah ditemukan. Stop.
2. Jika Q kosong, tidak ada solusi. Stop.
3. Jika Q tidak kosong, pilih dari antrian Q simpul i yang mempunyai nilai 'cost' ĉ(i) paling kecil. Jika terdapat beberapa simpul i yang memenuhi, pilih satu secara sembarang.
4. Jika simpul i adalah simpul solusi, berarti solusi sudah ditemukan, stop. Jika simpul i bukan simpul solusi, maka bangkitkan semua anak-anaknya. Jika i tidak mempunyai anak, kembali ke langkah 2.
5. Untuk setiap anak j dari simpul i, hitung ĉ(j), dan masukkan semua anak-anak tersebut ke dalam Q.
6. Kembali ke langkah 2.

Algoritma B&B dapat diaplikasikan pada beberapa persoalan NP-hard berikut:

- *Integer programming*
- *Nonlinear programming*
- *Travelling salesman problem (TSP)*
- *Quadratic assignment problem (QAP)*
- *Maximum satisfiability problem (MAX-SAT)*
- Pencarian tetangga terdekat
- *Cutting stock problem*
- *False noise analysis (FNA)*
- *Computational phylogenetics*
- Inversi himpunan
- Estimasi parameter
- *0/1 knapsack problem*
- Pemilihan fitur pada *machine learning*
- Prediksi terstruktur pada *computer vision*

III. Penyelesaian TSP Menggunakan Algoritma *Branch and Bound*

Persoalan TSP dapat dipecahkan dengan berbagai macam algoritma, salah satunya dengan algoritma B&B. B&B-TSP pun memilki berbagai macam cara penyelesaian. Yang akan dibahas di sini adalah B&B-TSP dengan bobot minimum tur lengkap dan matriks ongkos tereduksi dari graf.

A. Bobot Minimum Tur Lengkap

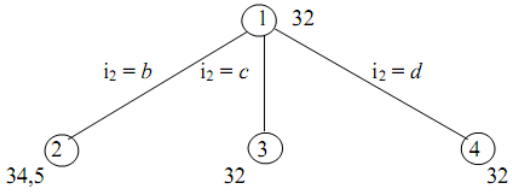
Dengan metode ini, setiap simpul harus dihitung bobot tur lengkapnya, yaitu:

$$\text{bobot tur lengkap} = 1/2 \sum_{i=1}^{n} \text{bobot sisi } i_1 + \text{bobot sisi } i_2 \quad (3)$$

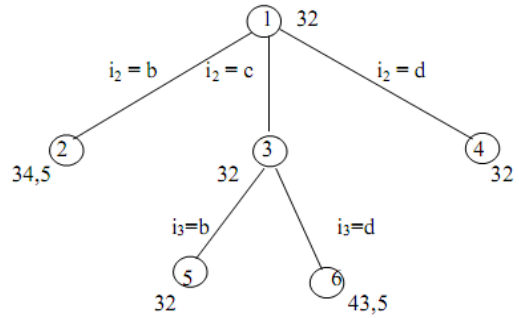
di mana sisi i_1 dan i_2 adalah dua sisi yang bersisian dengan simpul i di dalam tur lengkap. Sisi yang dipilih adalah sisi dengan bobot minimum, atau sisi yang jalurnya dilewati.

Setelah setiap simpul dihitung bobot tur lengkapnya, dicari nilai minimum dari setiap bobot tersebut untuk dijadikan sebagai fungsi pembatas. Simpul dengan bobot minimum tur lengkap lah yang akan diekspansi selanjutnya. Ketika mencapai solusi pertama, bobot tur tersebut dijadikan *the best solution so far*. Kemudian semua simpul yang bobotnya di atas bobot solusi pertama dibunuh. Pencarian dilakukan kembali pada simpul yang masih hidup.

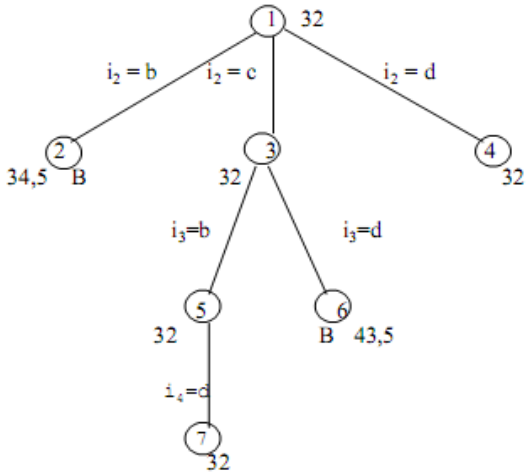
Sebagai contoh, Gambar 1 akan diselesaikan menggunakan metode ini. Berikut ini adalah tahapan pembentukan pohon ruang status yang terjadi.



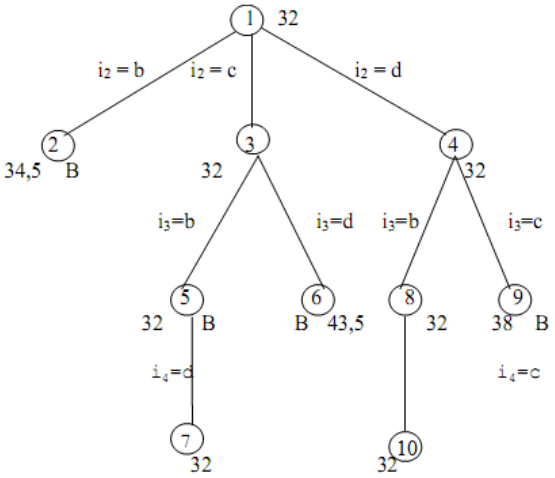
(a) Asumsi perjalanan dimulai dari simpul a. Dihitung bobot tur lengkap dari setiap simpul.



(b) Simpul 3 dan 4 memiliki bobot tur lengkap paling kecil, simpul 3 ditelusuri terlebih dahulu. Dihitung bobot tur lengkap ke simpul lain yang belum dikunjungi.



(c) Simpul 5 memiliki bobot tur lengkap terkecil, bobot 5 ditelusuri. Solusi pertama ditemukan, 32 disimpan sebagai *the best solution so far*. Simpul dengan bobot lebih besar dari 32 dibunuh.



(d) Simpul yang masih hidup ditelusuri, didapatkan solusi kedua.

Gambar 2 Contoh penerapan B&B-TSP dengan metode bobot minimum tur lengkap yang menghasilkan dua solusi, a-c-b-d-a dan a-d-b-c-a dengan bobot 32 (sumber: slide kuliah IF2211 Strategi Algoritma: Algoritma-Branch-&-Bound-(2016))

B. Matriks Ongkos Tereduksi

Sebuah matriks dikatakan tereduksi jika setiap kolom dan barisnya mengandung paling sedikit satu buah nol dan semua elemen lainnya non-negatif. Jumlah total elemen pengurang dari semua baris dan kolom merupakan batas bawah dari total bobot minimum tur[1].

Urutan langkah B&B-TSP dengan matriks ongkos tereduksi adalah sebagai berikut[1].

- Misalkan:
  - A: matriks tereduksi untuk simpul R.

- S: anak dari simpul R sehingga sisi (R, S) pada pohon ruang status berkoresponden dengan sisi (i, j) pada perjalanan.

- Jika S bukan simpul daun, maka matriks bobot tereduksi untuk simpul S dapat dihitung sebagai berikut:
  - (a) ubah semua nilai pada baris i dan kolom j menjadi ∞. Ini untuk mencegah agar tidak ada lintasan yang keluar dari simpul i atau masuk pada simpul j;
  - (b) ubah A(j, 1) menjadi ∞. Ini untuk mencegah penggunaan sisi (j, 1);
  - (c) reduksi kembali semua baris dan kolom pada matriks A kecuali untuk elemen ∞.
  - Jika r adalah total semua pengurang, maka nilai batas untuk simpul S adalah:

$$\hat{c}(S) = \hat{c}(R) + A(i, j) + r \qquad (4)$$

di mana,

$\hat{c}(S)$ = bobot perjalanan minimum yang melalui simpul S (simpul di pohon ruang status)

$\hat{c}(R)$ = bobot perjalanan minimum yang melalui simpul R, yang dalam hal ini R adalah orangtua dari S.

$A(i, j)$ = bobot sisi (i, j) pada graf G yang berkoresponden dengan sisi (R, S) pada pohon ruang status.

r = jumlah semua pengurang pada proses memperoleh matriks tereduksi untuk simpul S.

  - Hasil reduksi ini menghasilkan matriks B.

Agar lebih jelas, perhatikan contoh berikut. Pada Gambar 3, terdapat suatu matriks R yang merupakan representasi dari persoalan TSP. Matriks tersebut kemudian direduksi menjadi matriks A. Lalu pada matriks A dilakukan langkah-langkah di atas sampai akhirnya terbentuk sebuah pohon dan didapatkan hasil jalur dengan bobot terkecil.

R

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \longrightarrow \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

A

Gambar 3 Matriks R direduksi menjadi matriks A (sumber: slide kuliah IF2211 Strategi Algoritma: Algoritma-Branch-&-Bound-(2016))

IV. Optimasi Algoritma

B&B-TSP dapat dioptimasi kembali dengan beberapa cara berikut ini.

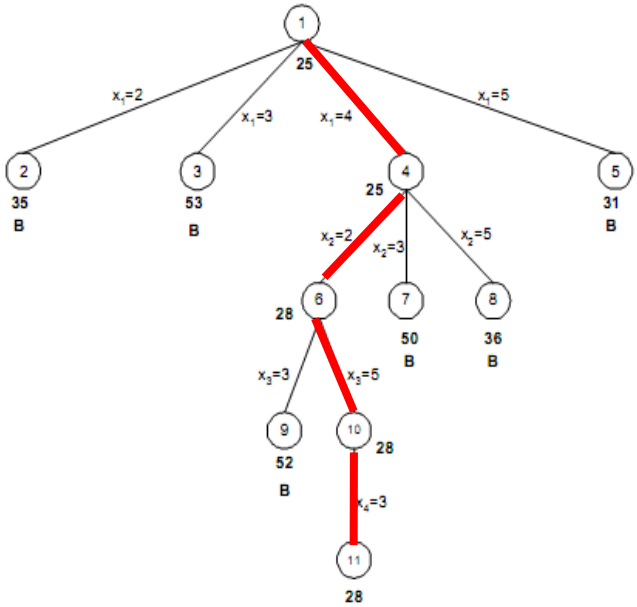
A. Cutting-plane Method

Algoritma B&B dapat diimplementasikan bersama dengan metode *cutting-plane* ini, disebut juga *branch-and-cut*. Algoritma ini dapat menyelesaikan persoalan sampai dengan 85.900 kota. Metode ini dipelopori oleh Ralph Gomory pada tahun 1950an sebagai metode untuk menyelesaikan persoalan *integer programming* dan *mixed-integer programming*. Tetapi saat itu parah ahli beranggapan bahwa metode tersebut masih tidak stabil dan tidak efektif.

Baru pada pertengah 1990an, Gerard Cornuejols dan rekan-rekannya menunjukkan bahwa metode *cutting-plane* ini menjadi sangat efektif ketika digabungkan dengan algoritma B&B.

Penyelesaian yang tepat untuk 15.112 kota di Jerman ditemukan pada tahun 2001 menggunakan metode ini. Komputasi ini dilakukan dengan jaringan 110 prosesor yang berlokasi di Rice University dan Princeton University. Kemudian pada bulan Mei 2004, TSP mengunjungi 24.978 kota di Swedia terselesaikan dengan panjang jalur 72.500 km, dan dapat dibuktikan bahwa tidak ada jalur lebih pendek yang memungkinkan.

Metode *cutting-plane* ini juga dapat diaplikasikan pada pemrograman nonlinear. Prinsipnya adalah dengan memperkirakan daerah yang *feasible* dari program linear dengan himpunan terbatas dan untuk menyelesaikan urutan perkiraan program linear.



Gambar 4 Pohon hasil penyelesaian B&B-TSP menggunakan matriks ongkos tereduksi dengan hasil jalur 1-4-2-5-3-1 dan bobot 28 (sumber: slide kuliah IF2211 Strategi Algoritma: Algoritma-Branch-&-Bound-(2016))

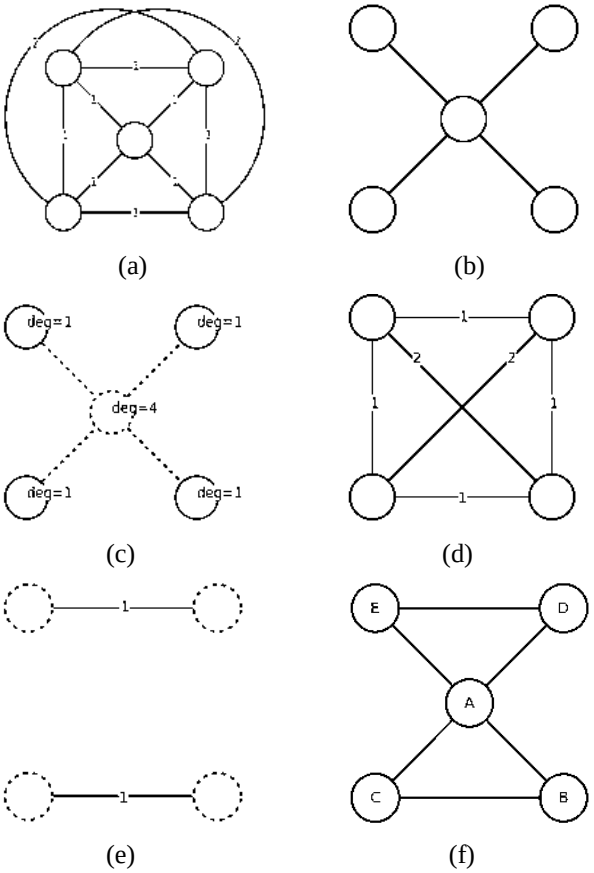
B. Algoritma Christofides

Algoritma Christofides adalah algoritma untuk memperkirakan solusi dari TSP metric, dengan jaminan solusi tidak lebih dari 3/2 dari panjang solusi optimal. Nama algoritma ini berasal dari penemunya, yaitu Nicos Christofides pada tahun 1976.

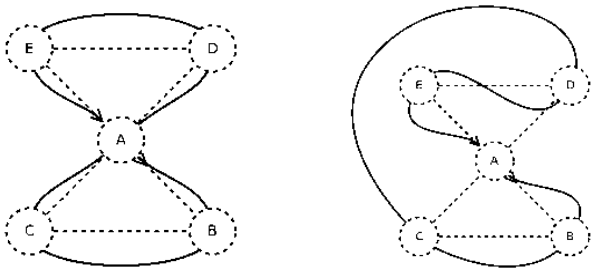
Algoritma ini dapat dituliskan seperti berikut.

1. Buat sebuah pohon merentang minimum *T* dari graf.
2. Misalkan *O* adalah himpunan simpul dengan derajat ganjil pada *T*. Dengan lemma jabat tangan, *O* memiliki memiliki jumlah simpul genap.
3. Cari sebuah sisi *M*, yaitu sisi dengan bobot minimum yang tidak memiliki simpul yang sama dengan sisi lain dalam graf yang dapat dibuat dari simpul-simpul pada *O*.
4. Satukan sisi dari *M* dan *T* untuk membentuk multigraf yang terhubung *H*, di mana setiap simpul berderajat genap.
5. Buat sebuah sirkuit Euler di *H*.
6. Ubah sirkuit dari tahap sebelumnya ke dalam sirkuit Hamilton dengan melewatkan simpul yang diulang.

Agar lebih jelas, perhatikan gambar berikut ini.



(a) (b) (c) (d) (e) (f)



(g) (h)

Gambar 5 Contoh penerapan algoritma Christofides (a) diberikan graf komplit yang bobot sisinya memenuhi ketidaksamaan segitiga, (b) kalkulasikan pohon merentang minimum, (c) kalkulasikan himpunan simpul dengan derajat ganjil, (d) bentuk sebuah subgraf menggunakan simpul-simpul berderajat ganjil tersebut, (e) buat sisi dengan bobot minimum yang tidak memiliki simpul yang sama dengan sisi lain, (f) satukan pohon merentang minimum dengan tahap sebelumnya, (g) kalkulasikan sirkuit Euler, (h) hilangkan simpul yang terulang

C. Ant Colony Optimization *(ACO)*

Ant Colony Optimization (ACO) adalah teknik yang digunakan untuk menyelesaikan permasalahan komputasional yang dapat direduksi untuk mencari jalur melalui graf. Algoritma ini pertama kali muncul dalam tesis PhD milik Marco Dorigo pada tahun 1992. Tujuan awal algoritma ini adalah untuk mencari jalur optimal dalam graf berdasarkan tingkah laku semut yang mencari jalur kepada koloninya dan sumber makanan.

Algoritma ini sudah banyak dikembangkan ke berbagai perluasan masalah. Tetapi pada awalnya memang algoritma ini digunakan untuk memecahkan TSP. Algoritma pertama ini disebut *the ant system*. Semut memiliki aturan ketika berpindah dari suatu kota ke kota lain, yaitu sebagai berikut.

1. Setiap kota hanya dikunjungi sekali.
2. Kota yang jauh mempunyai kemungkinan lebih kecil untuk dipilih.
3. Semakin jelas jejak feromon yang tersisa pada suatu sisi antara dua kota, semakin besar pula kemungkinan sisi tersebut dipilih.
4. Setelah menyelesaikan perjalanannya, semut akan menyimpan lebih banyak feromon pada sisi yang dilaluinya, jika perjalanannya pendek.
5. Setelah setiap iterasi, jejak feromon akan menguap.

V. KESIMPULAN

Travelling Salesman Problem (TSP) merupakan persoalan yang sampai sekarang masih dicari cara menyelesaikannya yang paling mangkus. Algoritma *Branch and Bound* adalah salah satu cara penyelesaian yang cukup populer. Namun algoritma ini masih dapat dimodifikasi agar lebih optimal dengan berbagai cara, di antaranya adalah *Ant Colony Optimization* (ACO), *cutting-plane method*, dan algoritma Christofides.

UCAPAN TERIMA KASIH

Pertama-tama saya mengucapkan puji dan syukur kepada Allah SWT. karena atas kehendak-Nya pembuatan makalah ini dapat berjalan lancar. Kemudian saya ucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulfah Maulidevi selaku pengajar mata kuliah Strategi Algoritma yang ilmunya digunakan sebagai dasar makalah ini. Terima kasih juga saya ucapkan kepada teman-teman yang telah menginspirasi topik bahasan melalui *brainstorming* serta senantiasa menyemangati selama pengerjaan makalah ini berlangsung.

REFERENSI

[1] Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Bandung: Informatika.

[2] Clausen, Jens (1999). Branch and Bound Algorithms—Principles and Examples (PDF) (Technical report). University of Copenhagen.

[3] Little, John D. C.; Murty, Katta G.; Sweeney, Dura W.; Karel, Caroline (1963). "An algorithm for the traveling salesman problem" (PDF). Operations Research 11 (6): 972–989.

[4] Applegate, D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J. (2006), The Traveling Salesman Problem, ISBN 0-691-12993-2.

[5] Arora, Sanjeev (1998), "Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems", Journal of the ACM 45 (5): 753–782.

[6] A. Colorni, M. Dorigo et V. Maniezzo, *Distributed Optimization by Ant Colonies*, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

[7] Goodrich, Michael T.; Tamassia, Roberto (2015), "18.1.2 The Christofides Approximation Algorithm", Algorithm Design and Applications, Wiley, pp. 513–514.

[8] Nicos Christofides, Worst-case analysis of a new heuristic for the travelling salesman problem, Report 388, Graduate School of Industrial Administration, CMU, 1976.

[9] Avriel, Mordecai (2003). Nonlinear Programming: Analysis and Methods. Dover Publications. ISBN 0-486-43227-0

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Mei 2016

Hasna Nur Karimah - 13514106