

ALGORITMA SEMUT PADA PENJADWALAN PRODUKSI JOBSHOP

Zainudin Zukhri, Shidiq Alhakim

Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia

Jl. Kaliurang Km. 14 Yogyakarta 55501

Telp. (0274) 895287 ext. 122, Faks. (0274) 895007 ext. 148

ABSTRAK

Algoritma semut merupakan salah satu dari sejumlah algoritma yang dapat diterapkan pada permasalahan optimisasi kombinatorial. Algoritma ini mengadopsi perilaku semut alamiah yang telah banyak diimplementasikan salah satunya pada penjadwalan produksi jobshop.

Penelitian ini memfokuskan pada implementasi penjadwalan produksi jobshop mesin dengan pendekatan algoritma semut, untuk menemukan fungsi tujuan, yaitu minimum makespan time dari seluruh kombinasi job-operasi yang didefinisikan.

Pada tulisan ini akan dipaparkan teknik pendekatan algoritma semut pada penjadwalan produksi jobshop yang kemudian diimplementasikan menjadi perangkat lunak yang akan diujikan dengan 5 job dan 5 mesin.

Keywords: Algoritma semut, Jobshop, graph, pheromone, tabu list

1. PENDAHULUAN

Perilaku sekumpulan semut dalam mencari atau menemukan sumber makanan dari sarangnya yang kemudian kembali lagi kesarangnya, merupakan inspirator ditemukannya algoritma semut oleh Dorigo, pada tahun 1991. Kemampuannya dalam menemukan solusi yang baik dari sekian solusi yang ada dikarenakan penggunaan informasi lokal berupa *pheromone* yang ditinggalkan setiap kali semut melewati suatu jalur. Semakin banyak semut yang melalui jalur tersebut maka *pheromone* yang ditinggalkan semakin banyak sehingga semut untuk menemukan jalur selanjutnya dengan menggunakan pemilihan acak yang disesuaikan berdasarkan besarnya *pheromone* yang dimiliki oleh jalur tersebut.

Implementasi algoritma semut telah banyak dilakukan seperti pada permasalahan TSP, *Spanning tree* dan penjadwalan. Pada permasalahan penjadwalan produksi *jobshop*, dalam penerapannya diperlukan teknik pendekatan *graph* yang akan menjadikan sebuah representasi dari urutan job-operasi dari kasus penjadwalan yang diberikan (van der Zwaan dan Marques, 1999).

Ada tiga jenis algoritma semut yaitu : *ant cycle*, *ant-density* dan *ant-quantity*. Yang membedakan antara ketiga jenis algoritma semut tersebut adalah pada proses pembarharuan *pheromone* semut. Pada penelitian ini pendekatan

algoritma semut yang digunakan adalah *ant cycle* untuk menemukan waktu penyelesaian keseluruhan job yang paling optimal.

2. METODOLOGI PENELITIAN

Tahapan yang dilakukan dalam penelitian ini adalah :

- Analisis masalah penjadwalan produksi *jobshop*
- Perancangan perangkat lunak untuk pemecahan masalah penjadwalan produksi dengan pendekatan algoritma semut.
- Implementasi dan hasil penelitian.

3. ALGORITMA SEMUT

Dalam algoritma semut terdapat beberapa parameter masukkan sebagai inialisasi awal untuk melakukan proses optimasi. Beberapa parameter tersebut adalah :

Parameter yang bersifat tetap :

- n = banyak kota yang ada.
- Q = Konstanta jumlah *pheromone*
- α = tetapan pengendali intensitas jejak semut
- β = tetapan pengendali visibilitas
- η_{ij} = visibilitas antar titik = $1 / d_{ij}$
- m = banyak semut
- ρ = tetapan penguapan jejak semut
- NC_{max} = jumlah siklus maksimum

3.1 Pemilihan Rute Perjalanan

Dalam kasus TSP pertama kali setiap semut disebar pada beberapa node secara random, sedangkan pada kasus *scheduling* , posisis awal semut terletak di posisi ke 0 yang selanjutnya semut-semut tersebut dibiarkan memilih rute perjalanannya (Purnomo, 2002). Untuk menjaga kelayakan dari solusi yang ditemukan, dan juga merupakan mekanisme pemanfaatan informasi jejak, maka semut-semut tersebut diatur untuk memilih sebuah node berikutnya dengan probabilitas ditentukan dengan formula :

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\frac{1}{d_{ij}}]^\beta}{\sum [\tau_{ij}(t)]^\alpha \cdot [\frac{1}{d_{ij}}]^\beta} \quad (1)$$

dimana

τ_{ij} = Intensitas *pheromone* pada jalur antara node(*job*)_i dan node(*job*)_j

d_{ij} = Jarak heuristik antara node(*job*)_i dan node(*job*)_j

p_{ij} = Probabilitas jalur dari node(*job*)_i ke node(*job*)_j

Persamaan 1 digunakan untuk dibandingkan dengan nilai yang dibangkitkan secara random, jika probabilitas node terpilih yang dituju

jumlahnya sudah lebih besar dibanding dengan nilai random yang dibangkitkan maka node tersebut dipilih sebagai node yang akan dikunjungi. Dan node tersebut akan disimpan sebagai node yang telah dikunjungi oleh semut, sehingga untuk menentukan node selanjutnya tidak akan dimasukkan dalam proses pencarian.

3.2 Pembaharuan Jumlah Jejak

Setelah semua semut telah melalui semua node yang tersedia, proses selanjutnya adalah melakukan pembaharuan jejak semut pada setiap jalur yang telah dilalui oleh semua semut, Berikut ini adalah rumusan untuk melakukan perhitungan pembaharuan jejak semut :

$$\tau_{ij}^{baru} = \rho \times \tau_{ij}^{lama} \times \Delta \tau_{ij} \quad (2)$$

dengan

$$\Delta \tau_{ij} : \begin{cases} \frac{Q}{\text{makespanti me}} & \text{jika node } i,j \text{ dilalui oleh semut.} \\ 0 & \text{untuk yang lainnya.} \end{cases}$$

Rumusan diatas akan memberikan sejumlah jejak semut dari hasil node yang telah dikunjungi oleh semut. Semakin banyak jumlah semut yang melalui jalur tersebut maka akan berbanding lurus dengan jumlah jejak semut. Dan juga semakin kecil jarak waktu antar node maka semakin kecil pula intensitas jejak semut yang akan ditambahkan pada jalur tersebut.

4. PENDEKATAN ALGORITMA SEMUT PADA PENJADWALAN JOB SHOP

Model yang digunakan untuk menotasikan masalah penjadwalan *jobshop* pada n -job dan m -mesin adalah (van der Zwaan dan Marques, 1999):

$$N / m / G / C \max \quad (3)$$

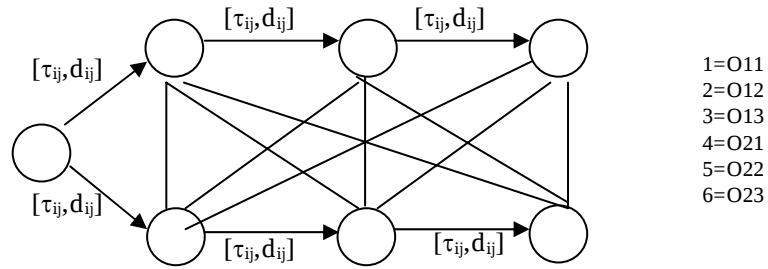
yang mana n mendefinisikan banyaknya *job* yang akan diproses, m menunjukkan banyaknya mesin yang dimiliki, $C_{\max}$ merupakan minimum *makspan time* dari suatu produksi dan G berisikan aturan urutan proses mesin untuk setiap *job* dan waktu prosesnya. Proses mesin dimodelkan dalam bentuk matrik T , semisal untuk T dengan $n=2$ dan $m=3$:

$$T = \begin{bmatrix} M1 & M2 & M3 \\ M2 & M3 & M1 \end{bmatrix} \quad (4)$$

Sedangkan untuk merepresentasikan waktu proses setiap operasi dimodelkan dengan matrik P :

$$P = \begin{bmatrix} t(O_{11}) & \dots & t(O_{1m}) \\ t(O_{21}) & \dots & t(O_{2m}) \\ \dots & \dots & \dots \\ \dots & \dots & \dots \\ t(O_{n1}) & \dots & t(O_{nm}) \end{bmatrix} \quad (5)$$

Untuk merepresentasikan permasalahan *jobshop* pada algoritma semut digunakan *graph* sebagai model yang akan mempermudah dalam pencarian minimum waktu penyelesaian. Berikut ini model *graph* dari 2/3/G/Cmax *jobshop*:



Gambar 1. Representasi dari 2/3/G/Cmax *job shop* pada *graph*

Setiap node dalam *graph* merupakan representasi dari *job* operasi pada matrik T, jumlah node dirumuskan dengan

$$Nodes = (n * m) + 1 \quad (6)$$

Sedangkan untuk banyaknya garis dinotasikan sebagai berikut

$$garis = \frac{|O| * (|O| - 1)}{2} + n \quad (7)$$

$$|O| = n * m \quad (8)$$

Dalam *graph* diatas setiap node memiliki batasan node yang dapat dikunjungi karena terdapat beberapa garis satu arah maka suatu node tidak memiliki hak untuk melalui jalur yang berlawanan arah dengan tanda arah panah tersebut. Dengan demikian setiap operasi penentuan kota harus terlebih dahulu memeriksa batasan kota yang boleh dilalui. Selain itu pemilihan node tujuan yang layak dilalui difilter dengan batasan operasi yang harus terurut. Misalkan jika *job* 1

operasi ke-2 hanya boleh dipilih sebagai node tujuan jika *job* 1 operasi ke-1 telah dipilih atau dilalui oleh semut.

4.1 Langkah Algoritma Semut Pada Job Shop

Langkah-langkah algoritma semut sebagaimana yang diungkapkan oleh Dorigo et al. (1996) dan Zukhri (2001), yang kemudian disesuaikan pada aplikasi penjadwalan produksi *jobshop*, yaitu :

1. Inisialisasi setiap parameter algoritma
2. Penyusunan rute kunjungan setiap semut ke setiap node
3. Perhitungan *makespan time* pada setiap semut
4. Pencarian rute dengan *makespan* terkecil
5. Perhitungan perubahan harga intensitas jejak kaki semut antar node .
6. Perhitungan harga intensitas jejak kaki semut antar node untuk siklus berikutnya.
7. Jika pemberhentian terpenuhi (kondisi konvergen pada rute yang dihasilkan dengan rute sebelumnya) atau jumlah max iterasi sudah selesai, ambil urutan *job*/operasi yang memiliki *makespan time* terkecil, jika tidak kembali ke langkah ke-2.

5. IMPLEMENTASI

Untuk mengimplementasikan permasalahan penjadwalan produksi *jobshop* digunakan bahasa pemrograman BORLAND DELPHI 5. Yang dapat dijalankan pada sistem operasi WINDOWS 9.X. Masukan sistem adalah urutan *job* dan waktu operasi dari setiap *job*. Karena sistem ini digunakan untuk penjadwalan mesin maka proses perakitan diabaikan.

6. HASIL PENGUJIAN DAN PEMBAHASAN

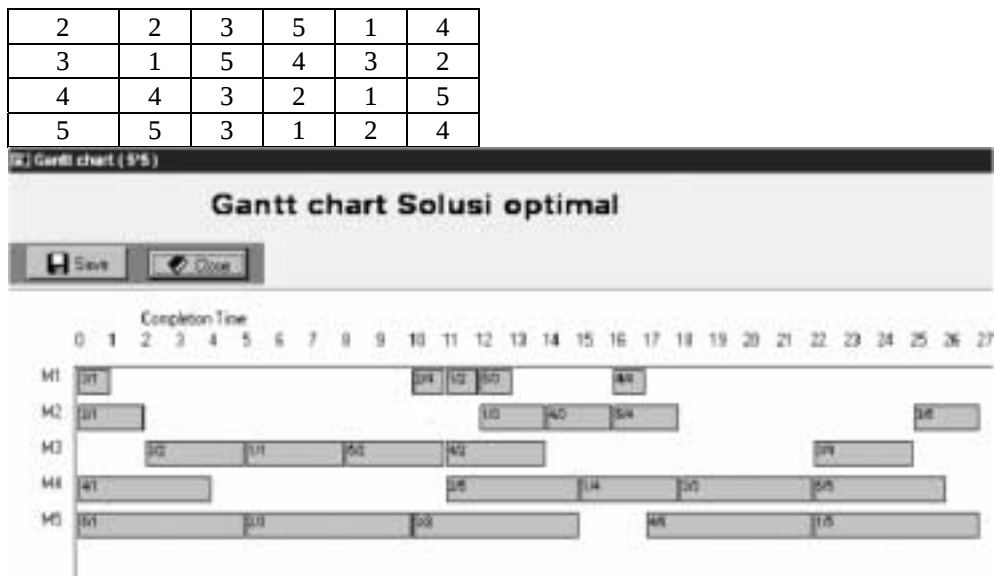
Untuk menguji perangkat lunak tersebut, diberikan suatu permasalahan penjadwalan dengan jumlah *job* mesin sebanyak 5 x 5 .

Tabel 1. Urutan mesin setiap *job*

	Operasi				
<i>Job</i>	1	2	3	4	5
1	3	1	2	4	5
2	2	3	5	1	4
3	1	5	4	3	2
4	4	3	2	1	5
5	5	3	1	2	4

Tabel 2. Waktu proses setiap *job*

	Operasi				
<i>Job</i>	1	2	3	4	5
1	3	1	2	4	5



Gambar 2. Gantt chart solusi optimal dengan AS

Parameter yang diberikan adalah : jumlah semut = 30, siklus = 5, $Q=1$, $\alpha=10$, $\beta=10$, $\rho=1$ dan $\tau=1$. Dari data tersebut didapatkan hasil *makespan time* sebesar 27 detik. Dengan membandingkan dari penelitian Agus ristono (Ristono, 2002) yang menggunakan pendekatan jaringan saraf tiruan didapatkan hasil *makespan time* sebesar 34 detik.

Dari perbandingan ini menandakan bahwa algoritma semut dapat menghasilkan urutan penjadwalan yang lebih optimal dengan interval 7 detik.

7. KESIMPULAN DAN SARAN

Berdasarkan analisis diatas, beberapa kesimpulan yang dapat dirumuskan adalah :

1. Algoritma semut dapat dijadikan pendekatan alternatif untuk menyelesaikan permasalahan penjadwalan produksi *jobshop*.
2. Sistem penjadwalan produksi *jobshop* dengan algoritma semut mampu memberikan usulan penjadwalan terbaik berdasarkan nilai minimal *makespan time*. Dan diharapkan usulan yang diberikan dapat menjadi masukan dalam proses pengambilan keputusan untuk melakukan proses produksi yang sebenarnya.
3. Pada penelitian ini algoritma semut masih diterapkan secara murni sehingga masih sangat memungkinkan untuk meningkatkan kinerjanya dengan hibridasi terhadap pendekatan lainnya.

PUSTAKA

- van der Zwaan, S., dan Marques, C. (1999). *Ant Colony Optimisation for Job Shop Scheduling*. Instituto Superior Tecnico.
- Purnomo, M. R. A. (2002). Hibridasi Algoritma semut dengan Algoritma Pencarian Lokal Pada Kasus Penjadwalan Flow Shop. *Makalah pada Seminar Nasional BKSTI III*, Surakarta.
- Dorigo, M., Maniezzo, V. dan Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. *IEE Trans. Syst, Man, Cyber*, 26(2), 29-41.
- Zukhri, Z. (2001) Penggunaan Dynamic Array pada Delphi 5 sebagai Tabu List dalam Algoritma Semut. *TEKNOIN*, 6(4), 291–303.
- Ristono, A. (2002). Artificial Neural Network in Jobshop Scheduling. *Prosiding Seminar Nasional*, Yogyakarta, 141-147.