Jurnal Teknik Industri, Vol. 11, No. 2, Desember 2009, pp. 188-194
ISSN 1411-2485

TABU SEARCH SEBAGAI *LOCAL SEARCH* PADA ALGORITMA *ANT COLONY* UNTUK PENJADWALAN *FLOWSHOP*

Iwan Halim Sahputra, Tanti Octavia, Agus Susanto Chandra
Fakultas Teknologi Industri, Jurusan Teknik Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya 60236
Email: iwanh@petra.ac.id, tanti@petra.ac.id

ABSTRAK

Ant colony optimization (ACO) adalah salah satu metode *meta-heuristic* yang dikembangkan untuk mencari solusi bagi permasalahan optimasi seperti penjadwalan. Metode *local search* merupakan salah satu bagian dari ACO yang menentukan kualitas solusi yang dihasilkan. Dalam makalah ini *tabu search* diusulkan sebagai metode *local search* dalam algoritma ACO untuk menyelesaikan masalah penjadwalan *flowshop*. Tujuan dari penjadwalan ini adalah untuk meminimalkan *makespan*. Hasil *makespan* dan *computation time* dari metode usulan dibandingkan dengan algoritma ACO yang menggunakan *job-index* sebagai metode *local search*. *Tabu search* sebagai *local search* menghasilkan nilai *makespan* yang tidak berbeda secara signifikan dibandingkan dengan menggunakan metode *job-index*, kelebihannya terletak pada *computation time* yang lebih singkat.

Kata kunci: *tabu search*, *ant colony algorithm*, *local search*, penjadwalan *flowshop.*

ABSTRACT

Ant colony optimization (ACO) is one of the meta-heuristic methods developed for finding solutions of optimization problems such as scheduling. Local search method is one part of the ACO which determines the quality of the resulting solution. In this paper, tabu search was proposed as a method of local search in ACO to solve the problem of flowshop scheduling. The purpose of this scheduling was to minimize the makespan. Makespan and computation time of the proposed method were compared to the ACO that implemented job-index as local search method. Using proposed algorithm, makespan values obtained were not significantly different tothe solutions of ACO using Job-Index method, but it had less computation time.

Keywords: *tabu search, ant colony algorithm, local search, flowshop schedulling.*

1. PENDAHULUAN

Tujuan dari dilakukannya penjadwalan produksi adalah untuk mencari waktu penyelesaian tersingkat atau meminimalkan *makespan*, meminimalkan keterlambatan dari batas waktu yang telah ditentukan (*due date),* dan meminimalkan mesin *idle*. Meskipun beberapa pendekatan eksak, seperti algoritma *branch-and-bound*, telah digunakan untuk menyelesaikan masalah penjadwalan secara optimal, pendekatan untuk mendapatkan solusi yang layak atau cukup baik dalam waktu yang lebih dapat diterima oleh pembuatan keputusan terus diteliti.

Ant colony optimization (ACO) adalah salah satu metode *meta-heuristic* yang dikembangkan untuk mencari solusi bagi permasalahan seperti penjadwalan. Colorni, Dorigo, dan Maniezzo (1991) memperkenalkan pertama kali algoritma ini. Stuetzle (1998) menggunakan algoritma ini untuk menyelesaikan masalah penjadwalan *flowshop* dengan tujuan untuk meminimalkan *makespan*. Merkle dan Middendorf (2000) menggunakannya untuk menyelesaikan masalah penjadwalan mesin dengan tujuan meminimalkan *tardiness*.

Lin, *et al.* (2008) menyatakan bahwa dalam ACO metode *local search* merupakan salah satu bagian dari ACO yang menentukan kualitas solusi yang dihasilkan. Rajendran dan Ziegler (2004) mengusulkan ACO (disebut PACO – *Proposed Ant-Colony Optimization*) yang menggunakan *job-index* sebagai metode *local search*. Metode yang mereka usulkan tersebut dilaporkan dapat menghasilkan solusi yang lebih baik dibandingkan dengan metode heuristik yang diusulkan oleh Liu dan Reeves (2001).

Tabu search (TS) adalah pendekatan meta-heuristik yang untuk pertama kalinya diusulkan oleh Glover (1986). Ben-Daya dan Al-Fawzan (1998) mengajukan metode berbasis TS, yang disebut BF-TS (Ben Fawsan-Tabu Search), untuk menyelesaikan masalah penjadwalan *flowshop*. Reeves (1995) membandingkan antara *genetic algorithm* dan *simulated annealing* untuk menyelesaikan masalah penjadwalan *flowshop*. Hasil yang didapatkannya adalah *simulated annealing* menghasilkan solusi yang lebih baik. Kinerja kombinasi *genetic algorithm* dan *tabu search* juga telah dibandingkan dengan *robust hybrid genetic* pada penjadwalan *flowshop* oleh Octavia, *et al.* (2007). Hasil penelitian tersebut menunjukkan kinerja kombinasi *genetic* dan *tabu search* memberikan solusi yang tidak berbeda. Berdasarkan kesimpulan dari makalah-makalah ini dapat dikatakan bahwa *tabu search* sama atau lebih baik dibandingkan dengan *simulated annealing* dan *genetic algorithm* dalam menyelesaikan permasalahan penjadwalan *flowshop*. Oleh karena itu, dalam makalah ini BF-TS yang dibangun oleh Ben-Daya dan Al-Fawzan (1998) diusulkan sebagai metode *local search* dalam algoritma PACO untuk menyelesaikan masalah penjadwalan *flowshop* dengan tujuan meminimalkan *makespan*. Hasil *makespan* dan *computation time* dari metode usulan ini akan dibandingkan dengan algoritma PACO untuk mengetahui performansinya.

2. METODE PENELITIAN

Penelitian ini dilakukan dengan membandingkan kinerja algoritma PACO dengan PACO-Tabu Search (PACO-TABU) yang diusulkan penulis. Perbedaan kedua algoritma tersebut ada pada metode *local search* yang digunakan yaitu *job index based* (lihat Lampiran: Gambar 1) untuk algoritma PACO (Lampiran: Gambar 2) dan algoritma BF-TS untuk algoritma *PACO-TABU* (Lampiran: Gambar 3).

Permasalahan yang digunakan pada penelitian ini adalah kombinasi penjadwalan *flowshop* dengan jumlah *job* 10, 20, 30, 40, dan 50 dengan 10, 20, dan 30 mesin. Algoritma dijalankan sebanyak 10 iterasi dengan masing-masing iterasi 100 kali pada masing-masing kombinasi jumlah *job* dan mesin. Rata-rata *makespan* dan *computation time* digunakan sebagai *performance measure* untuk perbandingan algoritma. Semakin kecil rata-rata *makespan* dan *computation time* yang dihasilkan semakin baik kinerja dari algoritma tersebut. Simulasi kedua algoritma menggunakan komputer dengan *processor* Intel Core2Duo 1,66GHz dan *memory* 1,5GHz.

Algoritma PACO-TABU bekerja dengan menggunakan solusi awal dari algoritma Nawaz, Enscore, dan Ham (NEH), algoritma BF-TS (Lampiran: Gambar 4) dan algoritma *ant colony*. Solusi awal yang didapat dari algoritma NEH ditingkatkan kinerjanya dengan menggunakan algoritma BF-TS. Peningkatan kinerja algoritma BF-TS diteruskan dengan algoritma *ant colony* hingga menghasilkan solusi terbaik pada saat *stopping criteria*.

Beberapa metode *local search* yang dapat digunakan pada algoritma BF-TS, yaitu:

1. *Swapping*, dilakukan dengan membangkitkan bilangan random *i* dan *j*. Bilangan random ini menunjukkan posisi *i* dan posisi *j*. Proses *swapping* bekerja dengan menukar *job* di posisi *i* dengan *job* di posisi *j*.
2. *Insertion*, proses pembangkitan bilangan random yang pada dasarnya hampir sama dengan metode *swapping*. Perbedaannya adalah *job* di posisi *i* dipindahkan ke posisi *j*.

3. *Block insertion*, proses ini dilakukan dengan membangkitkan bilangan *i*, *j*, dan *k* kemudian menyisipkan *k job* dimulai *job i* ke posisi *j*.

Algoritma BF-TS dilmulai dengan melakukan proses *neighborhood*. Beberapa proses *neighborhood* terakhir yang menghasilkan nilai optimum dimasukkan dalam *tabu list.* Proses *neighborhood* dilakukan dengan menggunakan metode *local search* yang terpilih. *Tabu list* yang ada berfungsi sebagai memori jangka pendek yang berguna menghindari pengulangan perhitungan. Ukuran *tabu list* yang ditetapkan pada penelitian ini sebesar 7, dengan ketentuan jika pasangan *job* dalam *tabu list* telah lebih dari 7 maka pasangan *job* pertama dikeluarkan dari *tabu list*. Ukuran *tabu list* yang terlalu kecil (misal 2) dapat menghasilkan kemungkinan solusi yang berulang sedangkan jika terlalu besar akan berpeluang menghasilkan *local optimum.* Algoritma pencarian ini juga menggunakan kombinasi *Intensification and Diversification Scheme* yaitu menggabungkan antara penggunaan atribut dari solusi-solusi yang didapat sebelumnya (*intensification scheme*) dengan penggalian solusi dari daerah yang belum pernah dijelajahi (*diversification scheme*). Proses pencarian yang telah dihasilkan akan dilanjutkan hingga memenuhi *stopping criteria* yang ditetapkan. *Stopping criteria* yang dipakai pada algoritma BF-TS adalah tidak terdapat perbaikan hasil pada suatu kriteria antara 2 solusi yang dihasilkan dari *diversification scheme* atau mencapai jumlah maksimum iterasi yang ditetapkan.

3. HASIL DAN DISKUSI

Perbandingan solusi yang diperoleh oleh algoritma PACO dengan PACO-TABU dilakukan untuk mengetahui performansi dari masing-masing algoritma dalam menyelesaikan permasalahan penjadwalan *flowshop.* Permasalahan yang digunakan adalah kombinasi penjadwalan dengan jumlah *job* 10, 20, 30, 40, dan 50 dengan 10, 20, dan 30 mesin. Untuk masing-masing kombinasi jumlah *job* dan mesin, algoritma dijalankan sebanyak 10 kali dan dihitung rata-rata *makespan* yang dihasilkan dan *computation time* yang diperlukan untuk mencapai solusi *makespan*. Kedua algoritma dijalankan pada komputer dengan *processor* Intel Core2Duo 1,66GHz dan *memory* 1,5GHz. Hasilnya dapat dilihat dari Tabel 1.

Tabel 1. Rangkuman hasil perbandingan PACO dan PACO-TABU

Jumlah *job* (*n*)	Jumlah mesin (*m*)	PACO		PACO-TABU	
		Makespan	*Computation time* (detik)	*Makespan*	*Computation time* (detik)
10	10	1099,8	2,31	1101,2	0,59
	20	1733,1	4,23	1733,3	1,44
	30	2369,4	6,52	2366,9	3,52
20	10	1642,5	34,80	1664,8	3,40
	20	2357,7	50,60	2371,0	6,10
	30	3006,7	61,40	3030,1	7,60
30	10	2203,0	76,00	2213,1	7,60
	20	2901,8	144,30	2920,3	12,80
	30	3607,5	211,00	3639,4	19,90
40	10	2746,7	166,70	2750,4	10,80
	20	3472,1	349,20	3533,1	17,90
	30	4232,3	514,60	4290,8	46,10
50	10	3206,7	347,50	3229,4	20,10
	20	4049,9	691,80	4102,9	50,90
	30	4798,2	1022,60	4853,4	69,30

Di Tabel 1 dapat dilihat bahwa algoritma PACO menghasilkan rata-rata *makespan* lebih baik daripada algoritma PACO-TABU. Hal ini karena metode *local search* pada algoritma PACO (*job index)* mencoba menghitung lebih banyak kemungkinan urutan *job* sehingga peluang mendapat urutan *job* dengan *makespan* yang lebih baik menjadi lebih besar. Untuk *computation time*, algoritma PACO-TABU lebih kecil rata-ratanya dibandingkan PACO. Hal ini disebabkan karena jumlah kemungkinan urutan *job* yang dihitung oleh algoritma PACO-TABU lebih sedikit daripada algoritma PACO.

Hal terburuk yang terjadi pada *job index* sebagai metode *local search* adalah mencoba kemungkinan alternatif urutan *job* sebanyak $3n^2$ kali. Sedangkan pada algoritma PACO-TABU yang menggunakan BF-TS sebagai *local search*, hal terburuk adalah mencoba kemungkinan urutan *job* sebanyak $4n$ kali. Dari sini dapat dilihat bahwa bila $n > 2$, algoritma PACO-TABU akan mencari lebih sedikit kemungkinan urutan *job* dari pada algoritma PACO.

Perbedaan rata-rata nilai *makespan* dan *computation time* antara algoritma PACO dengan PACO-TABU di Tabel 1 perlu diselidiki lebih lanjut apakah secara statistik berbeda secara signifikan dengan menggunakan $\alpha = 5\%$. Hasil uji varian dan *mean* menunjukkan tidak ada perbedaan yang signifikan pada semua kombinasi mesin untuk 10, 20, dan 50 *job* . Sedangkan, pada pengujian mean untuk kombinasi 40 *job* dengan 30 dan 40 mesin terdapat perbedaan yang signifikan dengan $\alpha = 5\%$. Uji *mean* dan varian untuk rata-rata *computation time* menunjukkan bahwa terdapat perbedaan yang signifikan untuk 20, 30, 40 dan 50 *job* dengan semua kombinasi mesin. Perbedaan yang tidak signifikan hanya terdapat pada kombinasi 10 *job* dengan 30 mesin. Ini berarti algoritma PACO-TABU menghasilkan solusi yang lebih cepat dalam hal waktu penyelesaian dengan hasil yang tidak berbeda secara signifikan dengan PACO.

4. KESIMPULAN

Berdasarkan analisa hasil solusi algoritma PACO dan PACO-TABU untuk menyelesaikan permasalahan penjadwalan *flowshop* dengan kriteria meminimalkan *makespan* dan *computation time*, dapat ditarik kesimpulan bahwa algoritma PACO-TABU menghasilkan nilai rata-rata *makespan* yangtidak berbeda secara signifikan dengan solusi dari algoritma PACO. Selain itu, rata-rata c*omputation time* untuk mencapai solusi *makespan* yang diperlukan oleh algoritma PACO-TABU lebih singkat dari pada algoritma PACO.Oleh karena itu, untuk mendapat solusi *makespan* bagi permasalahan penjadwalan *flowshop* yang layak dengan waktu yang relatif lebih cepat, maka algoritma PACO-TABU lebih disarankan dibandingkan PACO.

DAFTAR PUSTAKA

Ben-Daya, M., and Al-Fawzan, M. A., 1998. "A Tabu Search Approach for the Flowshop Scheduling Problem." *European Journal of Operational Research*, Vol. 109, pp. 88-95.

Colorni, A., Dorigo, M., and Maniezzo, V., 1991. "Positive Feedback as a Search Strategy." *Technical Report*, No. 91-016, Politecnico di Milano, Italy.

Glover, F., 1986. "Future Path for Integer Programming and Links to Artificial Intelligence." *Computers and Operations Research*, Vol. 13, No. 5, pp. 533-549.

Lin, B. M. T., Lu, C. Y., Shyu, S. J., and Tsai, C. Y., 2008. "Development of New Features of Ant Colony Optimization for Flowshop Scheduling." *International Journal of Production Economics*, Vol. 112, No. 2, pp. 742-755.

Liu, J., and Revees, C. R., 2001. "Constructive and Composite Heuristic Solutions to P||ΣCi Scheduling Problems." *European Journal of Operational Research*, Vol. 132, pp. 439-452.

Merkle, D., and Middendorf, M., 2000. "An Ant Algorithm with a New Pheromone Evaluation Rule for Total Tardiness Problems." *Computer Science*, Vol. 1803, pp. 287-296.

Nawaz, M., Enscore Jr., E. E., and Ham, I., 1983. "A Heuristic Algorithm for the m-Machine, n-Job Flowshop Sequencing Problem." *OMEGA*, Vol. 11, No. 1, pp. 91-95.

Octavia, T., Sahputra, I. H., and Soewanda, J., 2007. "Robust-Hybrid Genetic Algorithm for a Flow-Shop Scheduling Problem (A Case Study at PT FSCM Manufacturing Indonesia)." *Jurnal Teknik Industri*, Vol. 9, No. 2, pp. 144-151.

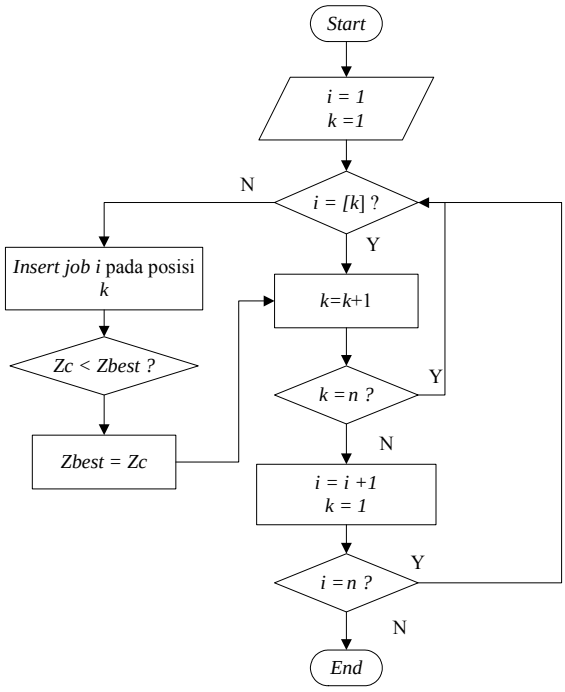
Ogbu, F. A., and Smith, D. K., 1990. "The Application of the Simulated Annealing Algorithm to the Solution of the n/m/Cmax Flow Shop Problem." *Computers and Operation Research*, Vol. 17, No. 3, pp. 243-253.

Rajendran, C., and Ziegler, H., 2004. "Ant-Colony Algorithm for Permutation Flowshop Scheduling to Minimize Makespan/Total Flowtime of Jobs." *European Journal of Operational Research*, Vol. 155, No. 2, pp. 426-438.

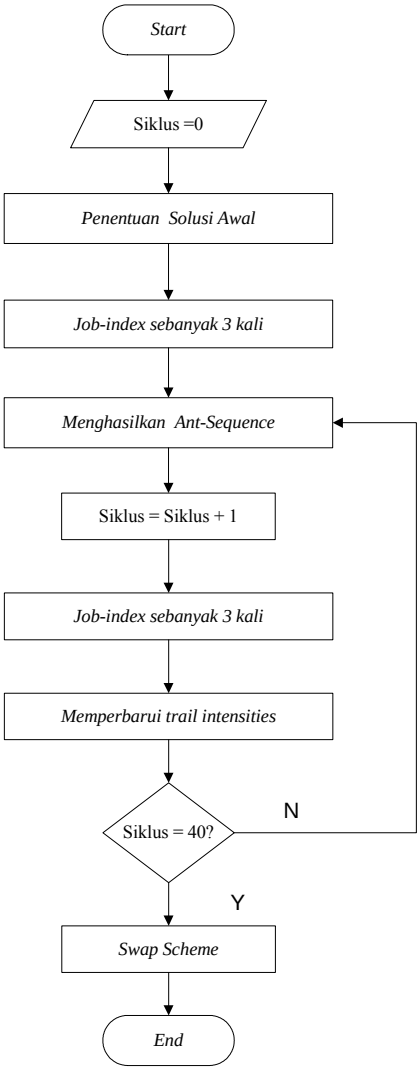
Reeves, C. R., 1995, "A Genetic Algorithm for Flowshop Sequencing." *Computers and Operation Research*, Vol 22, No. 1, pp. 5-13.

Stuetzle, T., 1998. "An Ant Approach for the Flowshop Problem." *Proceedings Of The 6th European Congress On Intelligent Techniques And Soft Computing*, Vol. 3., pp. 1560-1564, Verlag Mainz, Aachen, Germany.

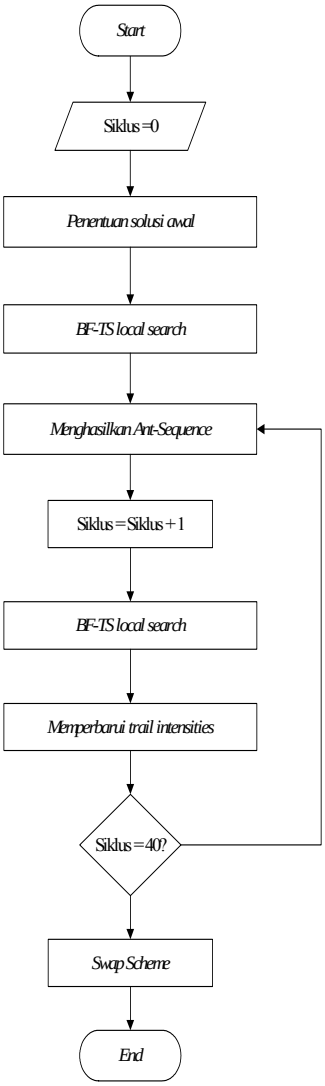
LAMPIRAN



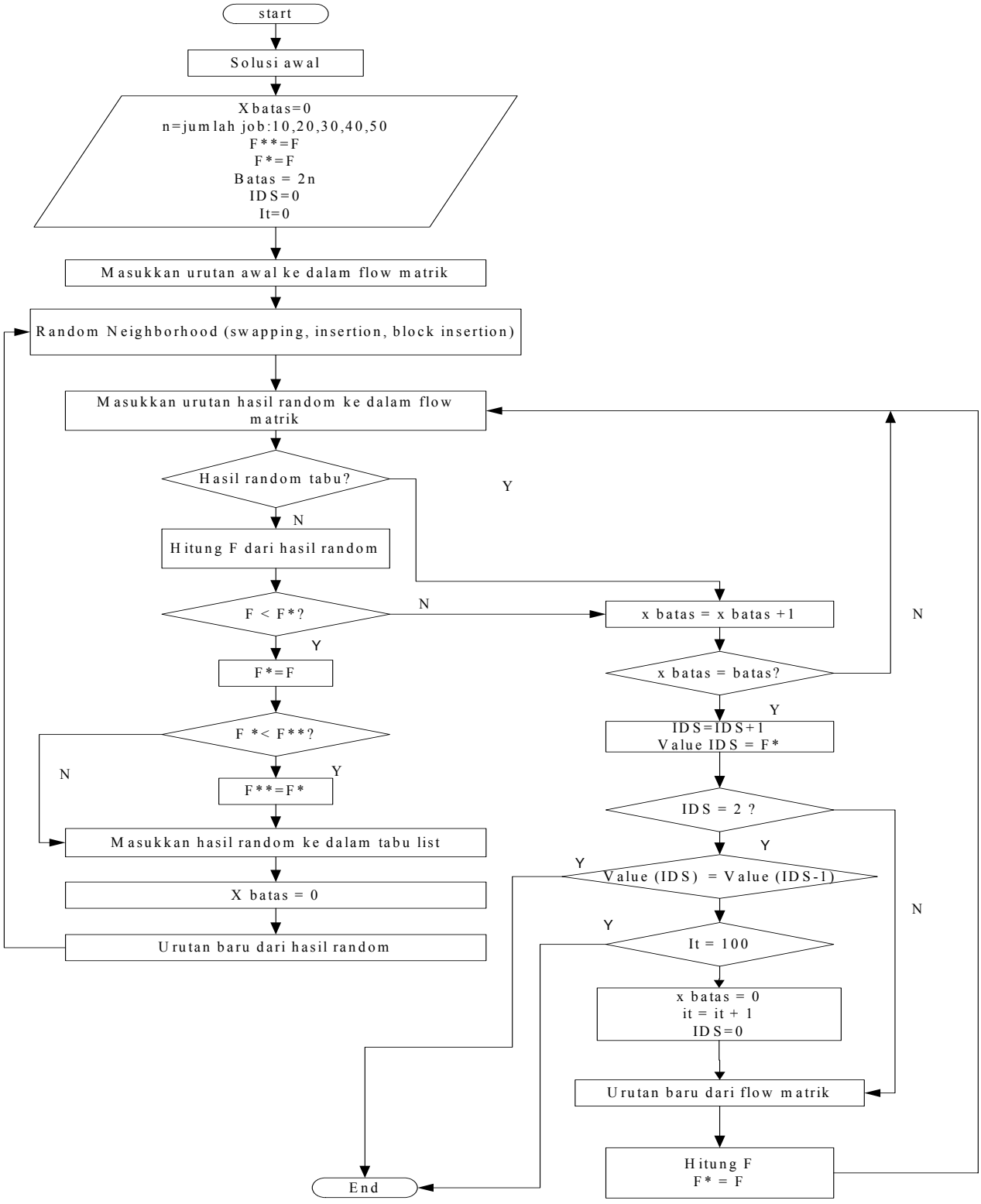
Gambar 1. *Flowchart* algoritma metode *job index* sebagai *local search*



Gambar 2. *Flowchart* algoritma PACO



Gambar 3. *Flowchart* algoritma PACO-TABU



Gambar 4. *Flowchart* algoritma BF-TS