STUDI PERBANDINGAN *PERFORMANCE* ALGORITMA HEURISTIK POUR TERHADAP *MIXED INTEGER PROGRAMMING* DALAM MENYELESAIKAN PENJADWALAN *FLOWSHOP*

Tessa Vanina Soetanto, Herry Christian Palit
Dosen Fakultas Teknologi Industri, Jurusan Teknik Industri-Universitas Kristen Petra
E-mail: tessa@petra.ac.id

Ika Munika
Alumni Fakultas Teknologi Industri, Jurusan Teknik Industri-Universitas Kristen Petra

ABSTRAK

Makalah ini menyajikan penelitian tentang *performance* algoritma heuristik Pour terhadap metode *Mixed Integer Programming (MIP)* dalam menyelesaikan masalah penjadwalan *flowshop* dengan tujuan meminimalkan *makespan*. Penilaian *performance* dilakukan berdasarkan nilai *Efficiency Index (EI), Relative Error (RE)* dan *Elapsed Runtime.*

Kata kunci: *flowshop, makespan*, algoritma heuristik Pour, *Mixed Integer Programming*.

ABSTRACT

This paper presents a study about new heuristic algorithm performance compared to Mixed Integer Programming (MIP) method in solving flowshop scheduling problem to reach minimum makespan. Performance appraisal is based on Efficiency Index (EI), Relative Error (RE) and Elapsed Runtime.

Keywords: *flowshop, makespan, Pour heuristic algorithm, Mixed Integer Programming.*

1. PENDAHULUAN

Masalah penjadwalan *flowshop* adalah menjadwalkan proses produksi dari masing-masing *n job* yang mempunyai urutan proses produksi dan melalui *m* mesin yang sama (Baker 1974, French 1982). Sudah banyak penelitian yang dilakukan untuk mencari penyelesaian yang optimal terhadap permasalahan ini seperti CDS (1970), Dannenbring (1977), NEH (1983) dengan tujuan meminimumkan waktu penyelesaian *job* (*makespan*).

Beberapa asumsi yang dipakai dalam penjadwalan *flowshop* adalah (1) tidak terdapat *preemption*, (2) semua *job* mempunyai *ready time* yang sama dan bersifat *independent* terhadap yang lain, (3) setiap mesin selalu siap untuk mengerjakan *job-job* yang ada tanpa adanya gangguan, seperti *machine breakdown* atau perawatan, dan (4) waktu *set up* bersifat *independent* dan termasuk dalam waktu proses.

2. ALGORITMA HEURISTIK POUR

Hamid Davoud Pour (2001) mengembangkan algoritma heuristik baru didalam menyelesaikan penjadwalan *flowshop* dengan tujuan meminalkan *makespan* yaitu berdasarkan pendekatan kombinasi. Hal ini dilakukan dengan cara mengganti setiap *job* dengan *job* yang lainnya dalam urutan sampai ditemukan kombinasi urutan yang dapat memenuhi kriteria tujuan.

Dalam metode ini diasumsikan bahwa semua *job* diproses secara terpisah dan *independent* untuk setiap mesinnya. Berikut adalah notasi yang digunakan:

- P_{ij} = waktu proses dari *job i* pada mesin *j*.
- C_{ij} =rentang waktu antara saat *job i* pada mesin *j* dimulai (*t=0*) sampai *job* itu selesai.
- *Ci = sum of completion time* untuk *job* i pada semua mesin.
- F_{max}=rentang waktu antara saat pekerjaan tersedia atau dapat dimulai sampai pekerjaan itu selesai (*makespan*).

Langkah-langkah pengerjaan Algoritma Heuristik Pour:

1) Memilih *job* secara acak sebagai urutan pertama sementara dalam urutan pengerjaan.
2) Menempatkan *job-job* lain (selain *job* yang sudah dipilih sebagai urutan pertama) pada urutan berikutnya.
3) Memilih waktu proses terkecil untuk masing-masing mesin.
4) Melakukan penambahan waktu proses secara increasing time pada P_{ij} yang lain, selain P_{ij} paling minimal yang terpilih sebelumnya.
5) Menghitung *sum of completion time (Ci*) untuk setiap *job* yang ada.
6) Mengurutkan *Ci* dengan aturan *increasing order* untuk diletakkan pada urutan setelah *job* yang sudah dipilih untuk urutan pertama sementara.
7) Setelah didapatkan urutan sementara, maka hitunglah *Fmax*-nya.
8) Melakukan ulang langkah 1-7 untuk setiap *job* yang ada sampai didapatkan *Fmax* paling minimal, yang akan ditempatkan sebagai urutan pertama dari urutan *job.*
9) Melakukan ulang langkah 1-8 sampai semua *job* berada pada urutan pengerjaan.

Berikut ini akan dijelaskan contoh pengerjaan algoritma *heuristic* Pour dengan menggunakan kasus kombinasi 5 *job* dan 5 mesin:

1) Memilih *job* 1 sebagai *job* pertama untuk ditempatkan dalam urutan pengerjaan.
2) *Job* 1 dipilih untuk menduduki urutan pertama sehingga waktu proses *job* 1 pada semua mesin dianggap 0. Tempatkan *job* selain *job* 1 sebagai urutan pertama pada urutan berikutnya.

Tabel 1. Data Waktu Proses Kombinasi 5 *job* dan 5 Mesin

	M1	M2	M3	M4	M5
J1	6	8	6	11	18
J2	17	8	6	9	8
J3	5	6	19	5	6
J4	8	10	8	17	9
J5	8	12	10	5	14

Tabel 2. *Job* 1 sebagai urutan pertama

	M1	M2	M3	M4	M5
J1	-	-	-	-	-
J2	17	8	6	9	8
J3	5	6	19	5	6
J4	8	10	8	17	9
J5	8	12	10	5	14

3) Memilih waktu proses terkecil untuk masing-masing mesin contoh *M1=5, M2=6* Lakukan penambahan waktu proses (*completion time*) pada setiap P_{ij} dengan aturan *increasing processing time*, contoh urutan waktu proses di *M1* terkecil-terbesar adalah *J3, J4, J5, J2* sehingga waktu proses *J4* di *M =5+8 =13, J5* di *M1=13+8=21, J2* di *M1=21+17=38.*
4) Hitunglah *sum of completion time (Ci*) untuk setiap *job* yang ada

Tabel 3. Langkah 3) dan 4)

	M1	M2	M3	M4	M5	Ci
J1	-	-	-	-	-	
J2	38	14	6	19	14	91
J3	5	6	43	5	6	65
J4	13	24	14	36	23	110
J5	21	36	24	10	37	128

5) dan 6) Dari Tabel 3 didapatkan urutan sementara dengan *job* 1 sebagai posisi pertama adalah *J1-J3-J2-J4-J5.*
7) Urutan *J1-J3-J2-J4-J5* mempunyai $F_{max} = 94$.

Tabel 4. Tabel Waktu Urutan J1-J3-J2-J4-J5

	M1	M2	M3	M4	M5
J1	0/6	6/14	14/20	20/31	31/49
J3	6/11	14/20	20/39	39/44	49/55
J2	11/28	39/36	39/45	45/54	55/63
J4	28/36	36/46	46/54	54/71	71/80
J5	36/44	46/58	58/68	71/76	80/94

8) Memilih *job* 2 sebagai urutan pertama sementara dalam pengerjaan dan langkah pengerjaan 1-7 berulang lagi. Dari semua urutan dengan *job* 1 sebagai posisi pertama, *job* 2 sebagai urutan pertama dan seterusnya dipilih urutan yang mempunyai F_{max} terkecil.
9) Dilakukan langkah 1-8 untuk *job* yang akan menempati posisi berikutnya.

3. MIXED INTEGER PROGRAMMING

Model Mixed Integer Programming (MIP) yang digunakan dalam penelitian ini dikembangkan oleh Blazewicz *et al.* (1996). Model MIP ini mempunyai fungsi tujuan yang sama dengan Algoritma heuristik Pour (2001) yaitu meminimalkan *makespan* untuk menyelesaikan masalah penjadwalan *flowshop*.

Notasi-notasi dalam MIP yang digunakan:
Indeks untuk *job* adalah *i, j = 1,…,n,* sedangkan untuk mesin *k, r = 1,…,m.*
Variabel keputusan:

$$Zij = \begin{cases} 1, & \text{bila job i terletak pada posisi j dalam permutasi} \\ 0, & \text{sebaliknya} \end{cases}$$

x_{jk} = *idle time* pada mesin *k* sebelum dimulainya *job* pada posisi ke-*j* di dalam permutasi.

y_{jk} = *waiting time* untuk *job* pada posisi ke-*j* dalam permutasi, setelah proses di mesin *k* selesai, sambil menunggu mesin *(k+1)* bebas untuk mengerjakan *job* selanjutnya.

F_{max} = *makespan* atau maksimum flow time dari seluruh *job* yang ada dalam urutan

Parameter:
P_{ri} = waktu proses dari *job* i pada mesin r
Fungsi tujuan: Meminimalkan F_{max}
Dengan kendala:

$$\sum_{j=1}^{n} z_{ij} = 1 \qquad i = 1, \ldots, n \tag{1}$$

$$\sum_{i=1}^{n} z_{ij} = 1 \qquad j = 1, \ldots, n \tag{2}$$

$$\sum_{i=1}^{n} p_{ri} z_{i,j+1} + y_{j+1,r} + x_{j+1,r} = y_{j,r} + \sum_{i=1}^{n} p_{r+1,i} z_{ij} + x_{j+1,r+1} \tag{3}$$

$$j = 1,\ldots,n-1 \qquad \text{r} = 1, \ldots, \text{m} - 1$$

$$\sum_{j=1}^{n} \sum_{i=1}^{n} p_{mi} z_{ij} + \sum_{j=1}^{n} x_{jm} = F_{max} \tag{4}$$

$$\sum_{r=1}^{k-1} \sum_{i-1}^{n} p_{ri} z_{i1} = x_{ik} \qquad \text{k} = 2, \ldots, \text{m} \tag{5}$$

$$y_{1k} = 0 \qquad \text{k} = 1, \ldots, \text{m} - 1 \tag{6}$$

Kendala (1) dan (2) digunakan untuk menempatkan *job* dan menentukan permutasi posisi dari *job-job* yang ada. Kendala (3) memberikan perhitungan Gantt Chart untuk semua pasangan mesin yang berdekatan dalam *flowshop* dengan m mesin. Kendala (4) digunakan untuk menghitung *maximum completion time* atau *makespan.* Kendala (5) digunakan untuk menghitung *idle time* dari mesin urutan kedua dna mesin selanjutnya, sementara menunggu kedatangan *job* urutan pertama. Kendala (6) digunakan untuk memastikan bahwa *job* urutan pertama dari permutasi selalu segera dikerjakan oleh mesin berikutnya.

4. PERFORMANCE PARAMETER

Dalam penelitian ini ada beberapa *performance* parameter yang digunakan untuk menentukan metode yang lebih baik yaitu:

Efficiency Index (EI) adalah perbandingan antara algoritma *heuristic* baru (Pour, 2001) dengan MIP, dirumuskan sebagai $EI = \frac{F_{max-MIP}}{F_{max-Pour}}$. Bila *EI =1* maka kedua metode memiliki performance yang sama, bila EI<1 maka algoritma heuristik Pour memiliki performance yang kurang baik dibanding dengan MIP. Demikian juga sebaliknya.

Relative Error (RE) digunakan untuk mengetahui seberapa jauh perbedaan *makespan* yang dihasilkan oleh kedua metode. $RE = \frac{F_{max-Pour} - F_{max-MIP}}{F_{max-Pour}} x100\%$.

Parameter *elapsed runtime* bertujuan untuk mengetahui seberapa jauh perbedaan waktu yang digunakan kedua metode untuk mendapat hasil akhir, dirumuskan dengan $elapsed\ \ runtime = time_{MIP} - time_{POUR}$

5. ANALISA PERBANDINGAN PERFORMANCE ALGORITMA HEURISTIK POUR TERHADAP MIP

Simulasi yang dilakukan untukmenyelesaikan permasalahan kombinasi *job* sejumlah 5–10 dengan jumlah mesin 5, 10, 15, 20 dan 25. Algoritma heuristik Pour dijalankan dengan bantuan *software Borland Pascal for Windows* sedangkan MIP diselesaikan dengan bantuan *software* simulasi LINGO. Kedua *software* tersebut menggunakan *personal computer (PC)* dengan klasifikasi Pentium 4, 1699 Mhz dan memory 128 MB. Adapun data dihasilkan secara acak dengan bantuan *software MINITAB 13.*

Adapun hasil perbandingan *makespan* antara kedua metode dan *performance* parameter disajikan dalam tabel berikut:

Tabel 5. Perbandingan *Makespan* antara Kedua Metode

Kombinasi		MIP		Pour		*Efficiency Index*	% Relative Error
Job	Mesin	Fmax	Runtime	Fmax	Runtime		
5	5	86	00:00:01	88	00:00:01	0.977	2.273
	10	165	00:00:01	169	00:00:01	0.976	2.367
	15	217	00:00:01	217	00:00:01	1	0
	20	331	00:00:01	333	00:00:01	0.994	0.6
	25	341	00:00:01	346	00:00:01	0.985	1.445
6	5	120	00:00:01	123	00:00:01	0.975	2.44
	10	209	00:00:01	213	00:00:01	0.98	1.878
	15	251	00:00:02	254	00:00:01	0.988	1.18
	20	319	00:00:03	325	00:00:01	0.981	1.85
	25	401	00:00:06	401	00:00:01	1	0
7	5	142	00:00:01	145	00:00:01	0.979	2.069
	10	203	00:00:02	205	00:00:01	0.99	0.976
	15	257	00:00:03	266	00:00:01	0.966	3.384
	20	318	00:00:15	334	00:00:01	0.95	4.79
	25	332	00:00:20	345	00:00:01	0.96	3.77

Lanjutan Tabel 5

8	5	145	00:00:01	165	00:00:01	0.878	12.12
	10	216	00:00:06	227	00:00:01	0.95	4.846
	15	270	00:00:22	280	00:00:01	0.96	3.57
	20	337	00:01:02	344	00:00:01	0.979	2.035
	25	406	00:02:24	432	00:00:01	0.9398	6.018
9	5	147	00:00:02	162	00:00:01	0.907	9.259
	10	204	00:00:11	213	00:00:01	0.9577	4.22
	15	277	00:01:35	282	00:00:01	0.98	1.77
	20	313	00:17:24	328	00:00:01	0.95	4.57
	25	420	00:27:02	444	00:00:01	0.945	5.4
10	5	159	00:00:01	167	00:00:01	0.95	4.79
	10	233	00:01:09	248	00:00:01	0.939	6.04
	15	321	00:37:52	326	00:00:01	0.984	1.534
	20	331	00:47:57	338	00:00:01	0.979	2.07
	25	412	08:50:42	424	00:00:01	0.97	2.83

Berdasarkan hasil simulasi yang terangkum di dalam Tabel 5 maka dapat dikatakan bahwa *performance* algoritma heuristik baru (Pour, 2001) sudah cukup baik dibandingkan dengan Mixed Integer Programming (MIP). Hal ini terlihat dari nilai *Efficiency Index (EI)* untuk algoritma heuristik Pour di semua kombinasi *job*-mesin kecuali kombinasi 8 *job*-5 mesin, berkisar diantara nilai 0.94 – 1.

Meskipun nilai *EI* yang dihasilkan tidak melebihi 1 tetapi dapat dikatakan bahwa *performance* dari algoritma heuristik baru sudah mendekati *performance Mixed Integer Programming* yang merupakan hasil optimal. Sedangkan ditinjau dari parameter *Relative Error (RE)* maka perbedaan antara algoritma heuristik Pour dengan *Mixed Integer Programming* tidak terlampau besar yang berada di sekitar lebih kecil dari 5%. Terdapat perkecualian untuk kasus tertentu yaitu kombinasi 8 *job* 25 mesin, 9 *job* 5 mesin, 9 *job* 25 mesin dan 10 *job* 10 mesin yang mempunyai RE diantara 5%-10%. Ditinjau dari segi parameter yang terakhir yaitu *Elapsed Runtime* maka terdapat perbedaan yang semakin meningkat seiring dengan bertambah kompleksnya permasalahan untuk *Mixed Integer Programming*. Hal ini tentu saja dikarenakan MIP merupakan metode optimasi yang membutuhkan lebih banyak waktu dalam penyelesaiannya dibandingkan dengan algoritma heuristik Pour.

6. KESIMPULAN

Algoritma heuristik Pour menunjukkan *performance* yang cukup baik dalam menyelesaikan permasalahan penjadwalan *flowshop* dengan tujuan meminimalkan *makespan* jika dibandingkan dengan salah satu metode optimasi *Mixed Integer Programming* (MIP). Hal ini terlihat dari hasil ketiga *performance* parameter yaitu *Efficiency Index* yang mendekati 1, *Relative Error* yang berkisar 5% *dan Elapsed Runtime* yang lebih singkat.

DAFTAR PUSTAKA

Blazewicz, J., K. H. Ecker, E. Pesch, Schmidt, J. Weglar, 1996. *Scheduling Computer and Mnufacturing Processes*, New York: Springer.

French, S., 1982. *Sequencing and Scheduling: an Introduction to the Mathematics of Job Shop*, Chichester: Ellies Horwood.

Pinedo, M., 2002. *Scheduling Theory, Algorithms, and Systems*, 2nd edition, Upper Saddle River, New Jersey: Prentice Hall.

Pour, H.D., 2001. "A New Heuristic for n-Job m-Machine Flowshop Problem", *Production Planning Control*, vol. 12, no.7, 648-653.

Baker, K. R., 1974. *Introduction to Squencing and Scheduling*, New York: John Wiley & Sons.

Stafford, E.F., 1988. "On The Development of a Mixed Integer Linear Programming Model for The Flowshop Sequencing Problem", *Journal Operation Research Society*, no. 39, 1163-1174.