R E M JURNAL
(REKAYASA ENERGI MANUFAKTUR)
R.E.M. (Rekayasa, Energi, Manufaktur) Jurnal Vol. 1 No. 2. 2016
ISSN 2527-5674 (print) , ISSN 2528-3723 (online)
Journal Homepage: http://ojs.umsida.ac.id/index.php/rem
DOI: http://dx.doi.org/10.21070/r.e.m.v1i2.404

Optimasi Penjadwalan *Flow Shop* Menggunakan Algoritma *Hybrid Differential Evolution*

Rudi Nurdiansyah *

Teknologi Industri, Universitas Negeri Malang

Article history: Received 11/10/2016, Revised 15/11/2016, Accepted 2/12/2016

ABSTRAK

Penjadwalan produksi merupakan bagian integral di dalam sistem manufaktur. Artikel ini menyelesaikan permasalahan penjadwalan *flow shop* dengan fungsi obyektif *total flow time*. Dalam penjadwalan, minimasi *total flow time* menghasilkan konsumsi yang stabil terhadap sumber daya, perputaran *job* yang cepat serta meminimalkan *work in process inventory*. Permasalahan penjadwalan *flow shop* tergolong pada permasalahan optimasi kombinatorial yang merupakan *NP-hard*. Saat ini, penggunaan algoritma metaheuristik banyak digunakan untuk memecahkan kasus optimasi kombinatorial, termasuk penjadwalan *flow shop*. Salah satu yang memiliki performa yang baik adalah Algoritma *Differential Evolution*. Untuk meningkatkan kualitas solusinya, Algoritma *Differential Evolution* akan ditambahkan dengan prosedur *local search*. Algoritma pada artikel ini yang dinamakan *Hybrid Differential Evolution*. Algoritma tersebut diuji menggunakan data penjadwalan *flow shop* yang ada pada *OR-Library*. Performa *Hybrid Differential Evolution* akan dibandingkan dengan *Hybrid* GA dan MOACSA. Hasil pengujian menunjukkan bahwa *Hybrid Differential Evolution* memberikan performa yang lebih baik dibandingkan dengan algoritma lain.

Keywords : Penjadwalan *flow shop*, *total flow time*, *hybrid differential evolution*

ABSTRACT

Scheduling is an integral part of advanced manufacturing systems. This article solves the flow shop scheduling problem with total flow time objective. In scheduling, total flow time minimization results in stable consumption of resources, rapid turn-around of jobs and work-in-process inventory minimization. Flow shop scheduling problem is combinatorial optimization and NP-hard. Recently, metaheuristics algorithms are widely used to solve combinatorial optimization, including flow shop scheduling. The one of the algorithms has good performance is Differential Evolution. To improve the quality of solution, Differential Evolution Algorithm in this article will combine with local search procedure. This algorithm is called Hybrid Differential Evolution. The proposed algorithm is tested with well-known problems in OR-Library. Its solution performance was compared with Hybrid GA and MOACSA. The computational results show that proposed algorithm is better than other methods compared.

Keywords : *Flow shop scheduling, total flow time, hybrid differential evolution*

Rudi Nurdiansyah lahir di Sidoarjo, 8 Nopember 1984. Menamatkan SD dan SMP di Sidoarjo, sedangkan SMA di Surabaya. Melanjutkan studi S1 di Universitas Muhammadiyah Sidoarjo pada Program Studi Teknik Industri dan lulus tahun 2008. Tahun 2009 melanjutkan S2 di Jurusan Teknik Industri Institut Teknologi Sepuluh Nopember Surabaya dengan bidang keahlian Sistem Optimasi Industri. Tahun 2012 mulai mengajar di Program Studi Teknik Industri Universitas Muhammadiyah Sidoarjo. Saat ini, penulis menjadi dosen di Jurusan Teknologi Industri Universitas Negeri Malang.

PENDAHULUAN

Penjadwalan produksi merupakan bagian integral di dalam sistem manufaktur [1]. Permasalahan di dalam penjadwalan adalah bagaimana mengalokasikan mesin produksi untuk melaksanakan serangkaian aktivitas penyelesaian *job* dalam waktu tertentu untuk mencapai suatu tujuan tertentu [2].

*Coresponding author.
E-mail address: rudi.nurdiansyah.ft@um.ac.id,Telp.(+62) 82231352349
Peer reviewed under reponsibility of Universitas Muhammadiyah Sidoarjo.

Penyelesaian *job* pada kebanyakan sistem manufaktur disusun secara berurutan [3]. Dalam penjadwalan, cara penyusunan *job* seperti itu disebut *flowshop*. Permasalahan penjadwalan *flowshop* memiliki ciri, *job* diproses dengan urutan yang sama pada paling tidak satu mesin dan satu mesin dapat memproses paling banyak satu *job* pada satu titik waktu. Tujuan yang ingin dicapai dari masalah ini, umumnya adalah meminimalkan total waktu penyelesaian keseluruhan *job* atau disebut *makespan*. Namun bisa juga mempertimbangkan tujuan lain seperti *total flowtime, total tardiness, total machine idle time* dan sebagainya.

Kebanyakan studi mengenai penjadwalan *flowshop* fokus pada minimasi *makespan* [4]. Pada kenyataannya, banyak tujuan lain selain *makespan* yang bisa dipertimbangkan sebagai obyektif seperti *total flowtime* yang juga merupakan ukuran kinerja yang sangat penting dalam meminimalkan ongkos penjadwalan total. Minimasi *total flow time* menghasilkan konsumsi yang stabil terhadap sumber daya, perputaran *job* yang cepat serta meminimalkan *work in process inventory* [4]. [5] menyelesaikan permasalahan permutasi penjadwalan *flow shop* dengan obyektif *total flow time* menggunakan *Hybrid Genetic Algorithm*. [6] menyelesaikan permasalahan penjadwalan juga menggunakan obyektif minimasi *total flow time*. Beberapa penelitian lain yang juga menggunakan obyektif yang sama antara lain [7],[8],[9] dan [10].

Permasalahan penjadwalan *flow shop* menjadi problem optimasi kombinatorial seiring dengan bertambahnya jumlah *job* dan jumlah mesin [11]. Problem optimasi kombinatorial merupakan *NP-hard* dan pendekatan yang lebih menjadi pilihan dari permasalahan ini adalah teknik solusi yang mendekati optimal menggunakan algoritma metaheuristik [12]. Pendekatan metaheuristik seperti *Simulated Annealing* (SA) [13], *Ant Colony Optimization* (ACO) [14], *Genetic Algorithm* (GA) [15], *Particle Swarm Optimization* (PSO) [16] dan *Artificial Immune Systems* (AIS) [17] dalam beberapa tahun terakhir banyak digunakan untuk memecahkan permasalahan optimasi kombinatorial karena terbukti memiliki kinerja komputasi yang baik [12].

Salah satu algoritma metaheuristik yang mempunyai reputasi sebagai metoda optimasi global optima yang efektif adalah *Differential Evolution* (DE). DE merupakan algoritma evolusioner yang efektif untuk pemasalahan optimasi kontinyu yang kompleks [18]. Keunggulan yang dimiliki oleh DE adalah konsep yang sederhana, implementasi yang mudah serta tingkat konvergensi yang cepat [19]. [20] menerapkan DE pada optimasi kombinatorial dan terbukti efektif. Penelitian lain juga menggunakan DE untuk menyelesaikan permasalahan optimasi kombinatorial seperti [21] dan [22].

Algoritma DE yang digunakan pada artikel ini akan diimprovisasi dengan menambahkan prosedur *local search*. Tujuannya adalah untuk meningkatkan kualitas solusi dari DE. [23] telah membuktikan bahwa DE dengan *local search* mempunyai performa yang lebih baik dibanding dengan *classic* DE. Beberapa penelitian lain, juga menggunakan *local search* untuk meningkatkan kualitas solusi dari DE, seperti [19], [24] maupun [25]. DE dengan penambahan prosedur *local search* pada penelitian ini, dinamakan *Hybrid* DE. Selanjutnya, algoritma tersebut akan digunakan untuk menyelesaikan permasalahan penjadwalan *flow shop* dengan obyektif *total flow time*. Untuk mengetahui performa dari algoritma yang digunakan pada penelitian ini, serangkaian percobaan akan dilakukan dengan menggunakan data dari [11]. Performa algoritma DE akan dibandingkan dengan *Hybrid Genetic Algorithm* (*Hybrid* GA) [5] dan *Multi-Objective Ant Colony System Algorithm* (MOACSA) [4].

Terminologi dan Formulasi Permasalahan Penjadwalan *Flow Shop*

Permasalahan penjadwalan *flow shop* memiliki asumsi, sejumlah *n job* (*n* = 1,2,3...i) dengan urutan yang sama dikerjakan pada serangkaian *m* mesin (*m* = 1,2,3...j) dengan waktu proses t_{ij}. Waktu proses t_{ij} merupakan waktu penyelesaian *job* ke-*i* pada mesin ke-*j*. Matriks penjadwalan *flow shop* dapat dilihat pada Tabel 1.

Tabel 1
Matriks Penjadwalan *Flow Shop*

Jobs	*Machines*						
	1	2	3	.	.	.	j
1	t_{11}	t_{12}	t_{13}	.	.	.	t_{1j}
2	t_{21}	t_{22}	t_{23}	.	.	.	t_{2j}
3	t_{32}	t_{32}	t_{33}	.	.	.	t_{3j}
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
i	t_{i1}	t_{i2}	t_{i3}	.	.	.	t_{ij}

Notasi yang digunakan adalah :

t_{ij} waktu proses *job* ke-*i* pada mesin ke-*j* (*i*=1,2,...,*n*), (*j*=1,2,...,*m*).
n jumlah *job* yang dijadwalkan
m jumlah mesin yang digunakan untuk memproses *job*
$\{\pi_1, \pi_2, ..., \pi_n\}$ permutasi *job*
S rangkaian urutan sejumlah *n job*
$C(\pi_i, j)$ waktu penyelesaian *job* π_i pada mesin *j*
F(S) *total flow time*

Total flow time dihitung dengan formula sebagai berikut :

$$F(S) = \sum_{i=1}^{n} C(\pi_i, m) \quad (1)$$

METODOLOGI

Algoritma *Differential Evolution* (DE) memiliki reputasi yang baik sebagai *global optimizer* yang efektif [26]. DE merupakan algoritma berdasarkan pembangkitan populasi. Algoritma yang digunakan untuk menyelesaikan permasalahan penjadwalan *flow shop* pada penelitian ini adalah *Hybrid* DE. Pada *Hybrid* DE, DE akan dikombinasikan dengan prosedur *local search*. Berikut ini adalah langkah-langkah *Hybrid* DE :

1. Inisialisasi

Sebelum melakukan inisialisasi populasi vektor variabel, terlebih dahulu menentukan batas atas dan batas bawah. Batas bawah dan batas atas digunakan sebagai langkah awal pembangkitan nilai variabel yang dicari. Untuk pembangkitan nilai awal variabel generasi ke-0, variabel ke-*j* dari vector ke-*i* direpresentasikan dengan notasi sebagai berikut :

$$x_{j,i,0} = lb_j + rand_j(0,1)(\ ub_j - lb_j\ ) \qquad (2)$$

Bilangan random dibangkitkan dengan fungsi *rand*, yang terletak antara [0,1]. Indeks *j* menunjukkan variabel ke-*j*. Hasil dari inisialisasi akan diberi indeks. Selanjutnya nilai tersebut akan diurutkan menggunakan prosedur SPV (*smallest position value*). Hasil pengurutan tersebut akan dijadikan solusi awal permutasi *job* berdasarkan indeks yang telah diberikan diawal.

2. Mutasi

Setelah inisialisasi, DE melakukan mutasi dan kombinasi terhadap populasi target untuk menghasilkan populasi percobaan dengan ukuran N vektor. Mutasi dilakukan dengan cara menambahkan perbedaan dua vektor (yang diambil secara acak) terhadap vektor ketiga dengan cara :

$$v_{i,g} = x_{r0,g} + F(x_{r1,g} - x_{r2,g}) \qquad (3)$$

Seperti terlihat pada persamaan di atas bahwa perbedaan dua vektor yang diambil secara acak, diskala terlebih dahulu sebelum ditambahkan ke vektor ketiga, $x_{r0,g}$. Faktor skala $F \in (0,1)$ bernilai riil positif yang berfungsi untuk mengendalikan tingkat pertumbuhan populasi. Meskipun tidak terdapat batas atas nilai *F*, nilai yang efektif adalah antara 0 dan 1. Indeks vektor basis, *r*0 dapat ditentukan dengan berbagai cara. Tetapi disini diasumsikan bahwa indeks *r*0 ditentukan secara acak. Selain indeks vektor berbeda satu sama lain dan berbeda dengan indeks vektor basis maupun vektor target, indeks *difference vector* (selisih antara $x_{r1,g}$ dan $x_{r2,g}$) juga dipilih sekali per mutasi.

3. Crossover

Untuk melengkapi strategi *differential mutation*, DE melakukan *uniform crossover*. Setiap vektor target $x_{i,g}$, dikawinsilangkan (*crossover*) dengan setiap vektor mutan $v_{i,g}$, sehingga membentuk vektor percobaan $u_{i,g}$ dengan formulasi :

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & if\ (rand_j(0,1) \le Cr, \quad or \quad j = j_{rand}) \\ x_{j,i,g} & sebaliknya \end{cases} \qquad (4)$$

Probabilitas *crossover*, $Cr \in [0,1]$ adalah nilai yang didefinisikan untuk mengendalikan fraksi nilai variabel yang disalin dari vektor mutan.

4. Local Search

Pada tahap ini, akan dilakukan prosedur *local search*. Prosedur *local search* yang digunakan adalah *insert-based local search*. Prosedur dari *insert-based local search* adalah sebagai berikut [19] :

Langkah 1: konversikan individu $X_{i,g}$ menjadi permutasi *job* π_{i_0} menurut aturan SPV.
Langkah 2: pilih secara acak *u* dan *v*, dimana $u \neq v$; $\pi_i = Insert\ (\pi_{i\_0}, u, v)$.
Langkah 3: tetapkan loop=1
Lakukan
Pilih secara acak *u* dan *v*, dimana $u \neq v$;
$\pi_{i\_1} = Insert\ (\pi_i, u, v)$;
jika $f\ (\pi_{i\_1}) < (\pi_i)$, maka $\pi_i = \pi_{i\_1}$;
loop++;
sampai *loop* < (*n*x(*n*-1))
Langkah 4: jika $f\ (\pi_i) < (\pi_{i\_0})$, maka $\pi_{i\_0} = \pi_i$;
Langkah 5: konversikan π_{i_0} kembali menjadi $X_{i,g}$

5. Seleksi

Jika populasi percobaan $u_{i,g}$, mempunyai fungsi tujuan lebih kecil dari fungsi tujuan populasi targetnya yaitu $x_{i,g}$, maka $u_{i,g}$ akan menggantikan posisi $x_{i,g}$ dalam populasi pada generasi berikutnya. Jika sebaliknya, target akan tetap pada posisinya dalam populasi.

$$X_{i,g+1} = \begin{cases} U_{i,g} & if(f(U_{i,g}) \le f(X_{i,g}) \\ X_{i,g} & sebaliknya \end{cases} \qquad (5)$$

6. Kriteria Pemberhentian

Kriteria pemberhentian yang digunakan adalah iterasi maksimal.

Algoritma *Hybrid* DE untuk menyelesaikan permasalahan penjadwalan *flow shop*

Prosedur dari *Hybrid* DE adalah sebagai berikut :

Langkah 1: Masukkan input parameter untuk *N*, *M*, $F \in [0,1]$ dan $Cr \in [0,1]$. Tetapkan $S = \phi$ dan batas bawah $(x_{j,i}) = -1$, sedangkan batas atas $(x_{j,i}) = 1$, $j = 1,...,N$.
Langkah 2: Inisialisasi populasi. Bangkitkan $X_{j,i,0} = lower(x_{j,i}) + random\ (0, 1)*(upper(x_{j,i}) - lower(x_{j,i})\ )$, $j = 1,...,N$ for $i = 1,...,M$.
Langkah 3: Konversikan individu $X_{j,i}$ menjadi permutasi *job* π_i berdasarkan aturan

SPV. Evaluasi setiap individu πi menggunakan *F(S)'s formula*.

Langkah 4 : Iterasi pertama g=1 .

Langkah 5: Lakukan mutasi and *Crossover*.
Pilih secara acak $r0$,$r1$, $r2 \in (1,...,M)$, dimana $r0 \neq r1 \neq r2 \neq$ i.
Hitung $V_{j,i}(g) = X_{r0,j,,i} + F^*(X_{r1,j,i} - X_{r2,j,i})$

Langkah 6: Jika *random*(0,1) ≤ Cr , maka $U_{j,i}(g) = V_{j,i}(g)$. Jika *random*(0,1) > Cr , maka $U_{j,i}(g) = X_{j,i}(g)$

Langkah 7: Aplikasikan *insert-based local search*

Langkah 7.1: Konversi individu $U_{j,i}(g)$ menjadi permutasi *job* π_{i_0} berdasarkan aturan SPV.

Langkah 7.2: Pilih secara acak u dan v, dimana $u \neq v$; $\pi_{i\_1} = Insert\ (\pi_{i\_0}, u, v)$.

Langkah 7.3: Tetapkan *loop*=1;
Lakukan
Pilih secara acak u dan v, dimana $u \neq v$;
$\pi_{i\_2} = Insert\ (\pi_{i\_1}, u, v)$;
jika $f(\pi_{i\_2}) < (\pi_{i\_1})$, maka $\pi_{i\_1} = \pi_{i\_2}$;
loop++;
sampai *loop* < (nx(n-1))

Langkah 7.4: jika $f(\pi_{i\_1}) < (\pi_{i\_0})$, maka $\pi_{i\_0} = \pi_{i\_1}$;

Langkah 7.5: konversikan kembali π_{i_0} menjadi $U_{j,i\ ,g}$

Langkah 8: Lakukan seleksi.
Jika $\pi_{i\_0} \leq \pi_i$, maka $X_{j,i,g+1} = U_{j,i}(g)$.
Jika $\pi_{i\_0} > \pi_i$, maka $X_{j,i,g+1} = X_{j,i}$

Langkah 9: Update S.

Langkah 10: Tetapkan $g = g+1$. jika $g < g_{max}$, maka kembali ke langkah 5.

Langkah 11: Output S dan nilai obyektif *total flow time*

HASIL

Performa dari *Hybrid* DE untuk menyelesaikan permasalahan penjadwalan *flow shop* dengan fungsi obyektif *total flow time* dievaluasi dengan melakukan simulasi komputer. *Hybrid* DE diuji pada kasus penjadwalan *flow shop* dengan data yang diunduh dari *OR-Library* (diakses pada 27 Juni 2016). Data yang digunakan pada penelitian ini adalah data kasus penjadwalan *flow shop* dengan 20 *job* dengan jumlah mesin bervariasi yaitu 5, 10 dan 20 mesin. Tabel 2 menunjukkan data yang digunakan pada penelitian ini.

Tabel 2
Data yang Digunakan

Job	Mesin	Kasus
20	5	ta001, ta002, ta005, ta008, ta009
20	10	ta011, ta014, ta016, ta018, ta020
20	20	ta022, ta024, ta025, ta026, ta028

Pengujian *Hybrid* DE dilakukan dengan membuat kode program algoritma pada *software* Matlab 7.8. Performa dari *Hybrid* DE dibandingkan dengan *Hybrid* GA [5] dan MOACSA [4]. Setiap pengujian dijalankan sebanyak 10 kali sehingga total 150 kali *running*. Solusi terbaik dipilih dari masing-masing kasus. Untuk mengevaluasi performa masing-masing algoritma, digunakan persentase relatif. Formulanya adalah sebagai berikut :

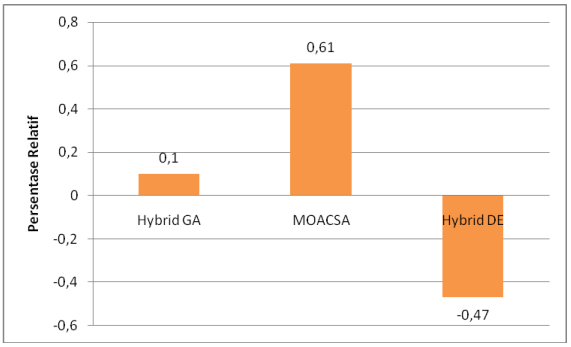
$$PR(S) = \frac{F(S) - \min F}{\min F} x\ 100\ \% \qquad (6)$$

PR(S) adalah persentase relatif. F(S) adalah total *flow time* masing-masing algoritma. Sedangkan min F adalah solusi terbaik yang ada pada literatur.

Persentase relatif dari masing-masing algoritma dapat dilihat pada Tabel 3. Hasil dari pengujian algoritma menunjukkan bahwa performa algoritma *Hybrid* DE lebih baik daripada *Hybrid* GA dan MOACSA. Hal ini dapat dilihat dari rata-rata persentase relatif yang lebih kecil dibanding yang lain. Bahkan pada beberapa kasus, performa *Hybrid* DE lebih kecil dari solusi terbaik yang ada pada literatur, ditandai dengan nilai negatif pada persentase relatif. Penambahan prosedur *local search* menggunakan *insert-based local search* pada algoritma DE terbukti mampu meningkatkan kualitas solusi yang dihasilkan DE. Gambar 1 adalah visualisasi perbandingan performa dari masing-masing algoritma.

Tabel 3
Persentase Relatif Masing-masing Algoritma

Kasus	*n x m*	Hybrid GA	MOACSA	Hybrid DE
ta001	20 x 5	0,00	0,96	0,41
ta002	20 x 5	0,00	0,00	0,42
ta005	20 x 5	0,00	0,38	-1,31
ta008	20 x 5	0,00	0,00	-1,46
ta009	20 x 5	0,00	0,00	-1,57
ta011	20 x 10	0,00	0,00	-0,55
ta014	20 x 10	0,13	0,00	-0,13
ta016	20 x 10	1,02	0,44	1,02
ta018	20 x 10	0,00	0,00	-0,66
ta020	20 x 10	0,00	0,60	1,37
ta022	20 x 20	0,00	0,00	0,17
ta024	20 x 20	0,00	0,05	0,26
ta025	20 x 20	0,00	5,89	-0,31
ta026	20 x 20	0,00	0,00	-2,48
ta028	20 x 20	0,00	0,00	1,16
Rata-rata		0,10	0,61	-0,47



Gambar 1 Perbandingan Performa 3 Algoritma

KESIMPULAN

Kasus penjadwalan *flow shop* banyak diteliti oleh para peneliti. Artikel ini menyelesaikan permasalahan penjadwalan *flow shop* menggunakan *Hydrid* DE. *Hybrid* DE adalah algoritma DE yang ditambahkan *prosedur local search* dalam pencarian solusi optimalnya.

Hasil pengujian *Hybrid* DE menunjukkan hasil persentase relatif yang lebih baik dari algoritma *Hybrid* GA maupun MOACSA. Dapat disimpulkan bahwa penambahan prosedur *local search* pada DE terbukti mampu meningkatkan kualitas solusi dari DE.

Ke depannya, algoritma yang diusulkan pada penelitian ini dapat digunakan untuk memecahkan kasus penjadwalan lain seperti *hybrid flow shop*, *no-wait flow shop* maupun *jos shop*. Selain itu, algoritma tersebut juga bisa diuji menggunakan fungsi obyektif yang lain seperti *makespan*, *total tardiness* ataupun *total machine idle time*.

REFERENSI

[1] Davendra D, Zelinka I, Bialic-Davendra M, Senkerik R, Jasek R. "Discrete self-organising migrating algorithm for flow-shop scheduling with no-wait makespan. Math Comput Model " 57:100–110. 2013

[2] Pinedo, M.L.: Scheduling: theory, algorithms, and systems. Springer. 2012.

[3] Javadi B, Saidi-Mehrabad M, Haji A, Mahdavi I, Jolai F, MahvadiAmiri N. "No-wait flow shop scheduling using fuzzy multiobjective linear programming". J Franklin Inst 345:452–467, 2008.

[4] Yagmahan, B. and Yenisey, M. M. "A multi-objective ant colony system algorithm for flow shop scheduling problem." *Expert Systems with Applications*, Vol. 37 No. 2, pp. 1361-1368. 2010.

[5] Yi Zhang, Xiaoping Li, Qian Wang. "Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization." European Journal of Operational Research Vol. 196. pp. 869–876. 2009.

[6] Gao, K. Z., Pan, Q. K., & Li, J. Q. "Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion". *The International Journal of Advanced Manufacturing Technology*, *56*(5-8), 683-692. 2011.

[7] Framinan, J.M., Leisten, R., Ruiz-Usano, R.,." Comparison of heuristics for flowtime minimisation in permutation flowshops". Computers and Operations Research 32, 1237–1254. 2005.

[8] Azizoglu M, Cakmark E, Kondakci S. "A flexible flowshop problem with total flow time minimization". Eur J Oper Res 132:528–538. 2001.

[9] Salmasi, N., Logendran, R., & Skandari, M. R. "Total flow time minimization in a flowshop sequence-dependent group scheduling problem". *Computers & Operations Research*, *37*(1), 199-212 2010.

[10] Jarboui, B., Eddaly, M., & Siarry, P. "An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems". *Computers & Operations Research*, *36*(9), 2638-2646. 2009.

[11] Taillard, E. "Benchmarks for basic scheduling problems", *european journal of operational research* Vol. 64 No.2 pp.278-285. 1993.

[12] Yagmahan, B. dan Yenisey, M. M. "Ant Colony Optimization for Multi-Objective Flow Shop Scheduling Problem", *Computers & Industrial Engineering*, Vol. 54, No. 3, hal. 411-420. 2008.

[13] Ishibuchi, H., Misaki, S. dan Tanaka, H., "Modified Simulated Annealing Algorithms for The Flow Shop Sequencing Problem", *European Journal of Operational Research*, Vol. 81, No. 2, hal. 388–398. 1995.

[14] Riyanto, O.A.W., & Santosa, B. "ACO-LS Algorithm for Solving No-wait Flow Shop Scheduling Problem. Intelligence in the Era of Big Data". Vol. 516 p. 89. 2015.

[15] Kordoghli, B., Jmali, M., Saadallah, S., dan Liouene, N., "Multi-Objective Scheduling of Flow Shop Problems in Finishing Factories using Genetic Algorithm", *Journal or Textile an Apparel, Technology and Management*, Vol. 6, No. 3, hal. 1-10. 2010.

[16] Lian, Z., Gu, X. dan Jiao, B. "A Similar Particle Swarm Optimization Algorithm for Permutation Flow Shop Scheduling to

Minimize Makespan", *Applied Mathematics and Computation*, Vol. 175, hal. 773–785. 2006.

[17] Gao, H. dan Liu, X. "Improved Artificial Immune Algorithm and Its Applications on Permutation Flow Shop Sequencing Problems", *Information Technology Journal*, Vol. 6, No. 6, hal. 929–933. 2007.

[18] Storn, R. dan Price, K. "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization Over Continuous Space", *Journal of Global Optimization*, Vol. 11, hal. 341-359.1997.

[19] Qian, B., Wang, L., Hu, R., Wang, W. L., Huang, D. X., and Wang, X. "A Hybrid differential evolution method for permutation flow-shop scheduling", *The International Journal of Advanced Manufacturing Technology*, Vol. 38 No.7-8 pp.757-777. 2008.

[20] Tasgetiren, M.F, Liang, Y.C, Sevkli, M, Gencyilmaz, G, "*Differential Evolution for Permutation Flowshop Sequencing Problem with Makespan Criterion"*, Dept. of Management, Fatih University, Istambul-Turkey. 2004.

[21] Pan, Q.K., Tasgetiren, M.F. dan Liang, Y.C. "A Discrete Differential Evolution Algorithm for The Permutation Flowshop Scheduling Problem", *Computers & Industrial Engineering*, Vol. 55, hal. 795–816. 2008.

[22] Mingyong, L. dan Erbao, C." An Improved Differential Evolution Algorithm for Vehicle Routing Problem with Simultaneous Pickups and Deliveries and Time Windows", *Engineering Applications of Artificial Intelligence*, Vol. 23, hal. 188–195. 2010.

[23] Noman, N. dan Iba H. "Accelerating Differential Evolution Using an Adaptive Local Search", *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 1. 2008.

[24] Sauer, J.G. dan Coelho, L. "Discrete Differential Evolution with Local Search to Solve the Traveling Salesman Problem: Fundamentals and Case Studies", *IEEE International Conference on Cybernetic Intelligent Systems*, London, hal. 1-6. 2008.

[25] Zamuda, A., Brest, J., Boskovic, B. dan Zumer, V. "Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization", IEEE Congress on Evolutionary Computation, Trondheim, hal. 195-202. 2009.

[26] Santosa, B. dan Willy, P. "*Metoda Metaheuristik : Konsep dan Implementasi*, " Guna Widya, Surabaya. 2011.

.