Algoritma Penjadwalan*Job Shop* Kelompok Mesin ParalelMenggunakan*Greedy Randomized Adaptive Search Procedure with Fixed Threshold* dengan Kriteria Minimisasi *Makespan*[*]

HABDHI VERDI USMAN, EMSOSFI ZAINI, ARIF IMRAN

Jurusan Teknik Industri
Institut Teknologi Nasional (Itenas) Bandung

Email: habdhiverdiusman@yahoo.com

ABSTRAK

Penelitian ini membahas tentang algoritma penjadwalan job shop kelompok mesin paralel menggunakan Greedy Randomized Adaptive Search Procedure(GRASP) with fixed threshold untuk minimisasi makespan. Pada metode ini, terdapat dua tahap untuk menyelesaikan permasalahan penjadwalan job shop kelompok mesin paralel. Tahap pertama merupakan tahap konstruksi untuk mendapatkan jadwal inisial. Tahap kedua merupakan tahap local search untuk memperbaiki jadwal inisial. Algoritma usulan diuji menggunakan set data dari literatur. Hasil yang didapat menunjukkan hasil yang sama baiknya dengan algoritma yang dikembangkan sebelumnya.

***Kata Kunci**: Penjadwalan Job Shop, Kelompok Mesin Paralel, GRASP, Threshold Accepting*

ABSTRACT

This paper discusses a job shop scheduling algorithms parallel machine groups using greedy randomized adaptive search procedure (GRASP) with a fixed threshold for makespan minimization. In this method, there are two phases to finish the job shop scheduling problem of parallel machine groups. The first phase is the phase of construction to obtain the initial schedule. The second stage is the phase local search to improve the initial schedule. Proposed algorithm was tested using data sets from the literature. The results showed equally good results with previously developed algorithms.

***Keywords**: Job Shop Scheduling, Parallel Machine Group, GRASP, Threshold Accepting*

[*] *Makalah ini merupakan ringkasan dari Tugas Akhir yang disusun oleh penulis pertama dengan pembimbingan penulis kedua dan ketiga. Makalah ini merupakan draft awal dan akan disempurnakan oleh para penulis untuk disajikan pada seminar nasional dan/atau jurnal nasional.*

1. PENDAHULUAN

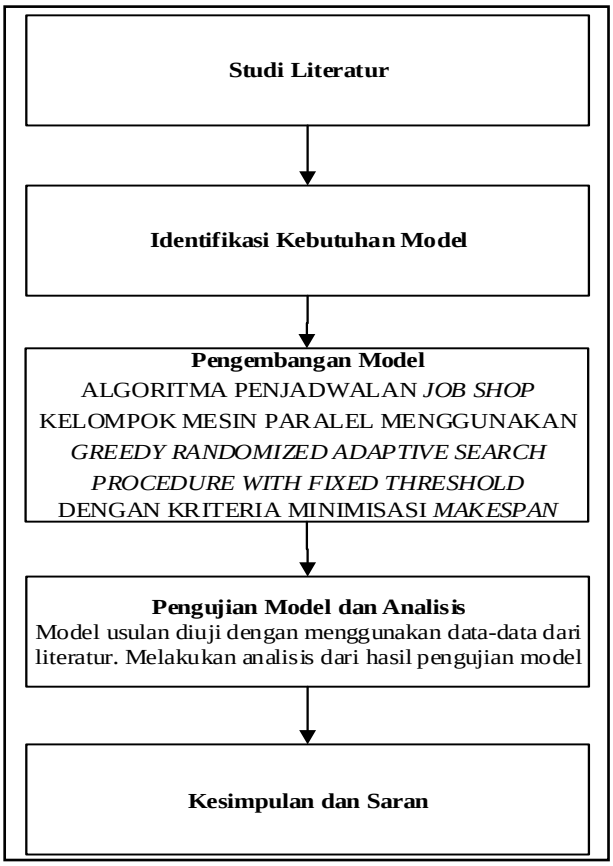
Penjadwalan *job shop* klasik (*Classical Job Scheduling Problem, CJSSP*) merupakan penjadwalan *njob* pada *m* mesin, dan setiap *job* memiliki *routing* (urutan mesin) masing-masing yang unik. Tujuannya adalah menghasilkan jadwal yang dapat meminimumkan waktu yang dibutuhkan untuk memproses seluruh *job* pada semua mesin (*makespan*). Permasalahan *job shop* termasuk ke dalam masalah *hard combinational optimization problems* atau yang umum dikenal sebagai *NP-Hard* (Garey dan Johnson, 1979). Teknik optimisasi berbasis *Branch and Bound*telah diusulkan untuk menyelesaikan CJSSP. Penelitian Carlier dan Pinson (1989) dan Brucker *et al.* (1994) menunjukkan bahwa teknik optimisasi dapat menghasilkan solusi optimal dalam waktu yang relatif pendek tetapi hanya untuk kasus-kasus kecil. Untuk CJSSP yang kompleks, teknik optimisasi menjadi tidak efisien karena waktu komputasi akan meningkat secara eksponensial seiring dengan pertambahan jumlah *job* ataupun mesin. Oleh karena itu, pendekatan berbasis heuristik dikembangkan untuk menyelesaikan permasalahan CJSSP yang kompleks. Pendekatan ini tidak menjamin menghasilkan solusi optimal tetapi dapat menghasilkan solusi dalam waktu yang relatif singkat.

Adams et al. (1998) membahas penjadwalan *job shop* menggunakan Algoritma *Shifting Bottleneck Heuristic* (SBH). penelitian tersebut mengasumsikan bahwa setiap stasiun kerja hanya dapat memproses satu *job* pada saat yang bersamaan, karena di setiap stasiun kerja hanya terdapat satu mesin. Sule dan Vijayasundaram (1998) mengusulkan metode heuristik dua fasa dalam menyelesaikan penjadwalan *job shop* kelompok mesin paralel homogen untuk minimasi makespan. Puryani (2003) membahas penjadwalan job shop kelompok mesin paralel heterogen menggunakan Algoritma jadwal aktif dan Algoritma *Branch and Bound Like* (BABL) dengan kriteria minimasi *makespan*. Scaparra dan Church (2005) menerapkan GRASP untuk menyelesaikan masalah transportasi di rural area. Putra (2010) telah membahas permasalahan job shop kelompok mesin paralel menggunakan GRASP untuk minimisasi *makespan*.

Threshold accepting dikembangkan oleh Dueck dan Scheuer (1990). *Threshold accepting* adalah sebuah metode pencarian solusi optimum kombinatorial dengan mengubah set *neighborhood*. Solusi baru diterima jika dan hanya jika berada dalam ambang batas yang telah ditentukan. Algoritma *Threshold accepting* memungkinkan solusi tidak terjebak pada *local search* minimum karena algoritma *Threshold accepting* menerima solusi baru yang mengarah ke nilai yang lebih tinggi. Memperhatikan uraian diatas, terlihat belum digunakan penggabungan algoritma GRASP dengan penggunaan *Fixed Threshold* untuk menyelesaikan penjadwalan *job shop* kelompok mesin paralel. Oleh karena itu, pada penelitian ini akan dikembangkan suatu algoritma penjadwalan *job shop* kelompok mesin paralel homogen dan heterogen menggunakan Algoritma GRASP with *Fixed Threshold* dengan kriteria minimisasi *makespan*.

2. METODOLOGI PENELITIAN

Metodologi penelitian bertujuan memberi penjelasan dan gambaran tentang tahapan dalam melakukan pengembangan algoritma. Tahapan-tahapan yang dilakukan dalam pengembanganalgoritma dapat dilihat pada Gambar 1.



Gambar 1. Tahapan Penelitian

Posisi penelitian ini terhadap penelitian sebelumnya dapat dilihat pada Gambar 2.

			Optimasi	*Heuristic*	*Metaheuristic*	
					GRASP	*GRASP with Fixed Threshold*
Job Shop	Mesin Tunggal		Carlier dan Pinson (1989), Brucker *et al.* (1994)	Adams *et al.* (1988), Toha dan Halim (1999), Wenqi dan Aihua (2004)	Binato *et al.* (2001), Alex *et al.* (2003)	
	Kelompok Mesin Paralel	Homogen		Sule dan Vijayasundaram (1998), Puryani (2003)	Putra (2010)	Penelitian
		Heterogen		Puryani (2003)		

Gambar 2. Peta Posisi Penelitian

3. PENGEMBANGAN ALGORITMA

Notasi-notasi yang digunakan dalam algoritma penelitian adalah:

K = iterasi.
N = Jumlah *job*.
O = Operasi.
p = Jumlah operasi.
q = Jumlah maksimum *r*.

S_r = Himpunan operasi-operasi yang siap dijadwalkan pada tahap ke-r. (r=1,2,...q).
O_{ijkm} = Operasi j, *jobi* stasiun kerja k di mesin paralel m.
C_{ijkm} = Saat paling awal *job* i operasi j dapat diselesaikan dengan stasiun kerja k di mesin paralel m.
σ_{ijkm} = Saat paling awal *job* i operasi j dengan stasiun kerja k di mesin paralel m.
t_{ijkm} = Waktu proses *job* i operasi j dengan stasiun kerja k di mesin paralel m.
$g(O)$= Fungsi *greedy* minimum, yaitu operasi-operasi O_{ijkm} yang dijadwalkan.
g_{min} = *Completion time* minimum dari operasi-operasi O_{ijkm} yang telah dijadwalkan.
g_{max} = *Completion time* maksimum dari operasi-operasi O_{ijkm} yang telah dijadwalkan.
L_{min} = Jumlah minimum kandidat elemen di dalam RCL.
O^* = Operasi O_{ijkm} yang terpilih secara *random* dari dalam RCL untuk dijadwalkan.
$PM_{[h]}$ = Operasi yang mendahului operasi h pada mesin paralel yang sama.
$PJ_{[h]}$ = Operasi yang mendahului operasi h untuk *job* yang sama.
$PJ[PM_{[h]}]$ = Operasi yang mendahului $PM[h]$ untuk *job* yang sama.
$SM_{[h]}$ = Operasi yang diproses setelah operasi h pada stasiun kerja dan mesin paralel yang sama.
$\Omega_{PM[h]}$ = Himpunan operasi-operasi yang mendahului operasi h untuk mesin paralel yang sama.
$\Omega_{PJ[h]}$ = Himpunan operasi-operasi yang mendahului operasi h untuk *job* yang sama.
$\Omega_{PJ[PM[h]]}$ = Himpunan operasi-operasi yang mendahului operasi $PM[h]$ untuk *job* yang sama.
Ω_{RS} = Himpunan operasi-operasi yang perlu dijadwalkan kembali.
$R_{PM[h]m}$ = *Ready time* operasi $PM[h]$ pada mesin paralel m.
$t_{PM[h]m}$ = Waktu proses operasi $PM[h]$ pada mesin paralel m.
$R_{PJ[h]}$ = *Ready time* operasi $PJ[h]$.
$t_{PJ[h]}$ = Waktu proses operasi $PJ[h]$.
$R_{[h]m}$ = *Ready time* operasi h pada mesin paralel m.
R_{km} = *Ready time* stasiun kerja k mesin paralel m.
$C_{[h]}$ = *Completion time* operasi h.
$\%_{Th}$ = Persentase batas *threshold*.
MS_{Th} = Batas atas nilai *makespan* berdasarkan $\%Th$.
MS = *Makespan*.
MS_e = Makespan dalam iterasi.
e = Jumlah tahap dalam iterasi.
W = Himpunan pasangan operasi yang dipertukarkan.
X = Jumlah pasangan operasi yang dipertukarkan.
Y = Pasangan operasi ke-Y, (Y= 1,2,...X) dalam W yang akan dipertukarkan.
W' = Himpunan pasangan operasi yang dipertukarkan dalam iterasi.
X' = Jumlah pasangan operasi yang dipertukarkan dalam iterasi.
Y' = Pasangan operasi ke-Y', (Y'= 1,2,...X') dalam W' yang akan dipertukarkan.
V = Jumlah operasi dalam Ω_{RS}.
S = Jumlah *job* yang memiliki sisa operasi terbanyak.
F = Jumlah operasi yang telah terjadwalkan dalam Ω_{RS}.

Langkah-langkah algoritma GRASP *with fixed threshold* dalam menyelesaikan masalah penjadwalan *job shop* kelompok mesin paralel dengan kriteria minimisasi *makespan* adalah:

Langkah 1: Set K=1
maxiter_G = maks (2,$\lceil \boldsymbol{n}/\mathbf{3} \rceil$)
Langkah 2: Bangkitkan jadwal inisial dengan menggunakan algoritma tahap konstruksi.
Langkah 3: Perbaiki jadwal inisial dengan menggunakan algoritma tahap *local search*.

Langkah 4: *Set K = K+*1 dan periksa apakah *K > maxiter_G* ?
Jika ya, pilih jadwal yang menghasilkan *makespan* terbaik.
Jika tidak, kembali ke langkah 2.

TAHAP KONSTRUKSI (Jadwal Inisial)
Langkah 1: Input data matriks *routing* dan matriks waktu proses.
Set L_{min} dengan persamaan L_{min} = maks *(2,*$\lceil \boldsymbol{n/3} \rceil$)
Langkah 2: Set *r*=1
Langkah 3: Tentukan S_r
Langkah 4: Hitung C_{ijkm} dengan menggunakan persamaan $C_{ijkm} = \sigma_{ijkm} + t_{ijkm}$
Langkah 5: Bangkitkan $\boldsymbol{\alpha} \in [\mathbf{0}, \mathbf{1}]$
Langkah 6: Bentuk RCL dengan menggunakan persamaan
$RCL = \{O \in S_r \mid g_{min} \leq g(O) \leq g_{min} + \boldsymbol{\alpha}\ (g_{max} - g_{min})\}$
Langkah 7: Periksa apakah $\left|\text{RCL}\right| \geq L_{min}$?
Jika ya, lanjutkan ke langkah 8.
Lainnya maka urutkan C_{ijkm} dalam $S_r$dari mulai yang terkecil sampai terbesar dan bentuk RCL dari L_{min}urutan pertama, kemudian pilih sebanyak L_{min} untuk masuk ke dalam RCL.
Langkah 8: Pilih satu operasi dalam RCL secara *random*, nyatakan sebagai *O**, dan jadwalkan.
Langkah 9: Untuk setiap solusi baru yang dihasilkan pada Langkah 8, perbaharui data:
a. Keluarkan operasi yang telah terjadwalkan dari dalam S_r,
$S_r = S_r - \{O^*\}$
b. *Set r = r+*1
c. Masukkan operasi berikutnya dari *job* yang sama ke dalam S_r.
Langkah 10: Jika $S_r \neq \{\}$, kembali ke Langkah 3.
Lainnya, proses pembentukan jadwal inisial selesai.
Langkah 11: Hitung *makespan* dengan menggunakan persamaan
MS = maks $\{C_{ijkm}\}$
Langkah 12: Tentukan nilai MS_{Th} dengan menggunakan persamaan:
$MS_{Th} = MS + (MS\ x\ \%_{Th})$

TAHAP LOCAL SEARCH (Perbaikan Solusi)
Langkah 1: Set *iter* = 1
Maxiter = maks (3,$\lceil \boldsymbol{n/X} \rceil$)
Input jadwal inisial dari tahap konstruksi.
Langkah 2: Set MS_o
Tentukan lintasan kritis. Kemudian pilih satu lintasan yang memberikan pertukaran pasangan operasi terbanyak, set *LK*.
Langkah 3: Tentukan *W*, Hitung *X*
Langkah 4: Set *Y* = 1
Langkah 5: Hitung *makespan* untuk operasi yang dipertukarkan menggunakan sub Algoritma Pertukaran Dua Operasi Pada Lintasan Kritis.
Tampilkan *gantt chart* dari dua operasi yang dipertukarkan.
Langkah 6: Apakah $MS \leq MS_{Th}$?
Jika ya, ke langkah 7.
Lainnya, ke langkah 12.
Langkah 7: Tentukan Lintasan Kritis, Tentukan *W'*
Hitung *X'*
Langkah 8: Set *Y'* = 1

Langkah 9: Hitung *makespan* untuk operasi yang dipertukarkan menggunakan sub Algoritma Pertukaran Dua Operasi Pada Lintasan Kritis. Nyatakan makespan dengan MS_e.

Langkah 10: Apakah $MS_e \leq MS_{Th}$?
Jika ya, ke langkah 11.
Lainnya, abaikan.

Langkah 11: Set $Y' = Y' + 1$, Apakah $Y' > X'$?
Jika ya, ke Langkah 12.
Lainnya, kembali ke Langkah 9.

Langkah 12: Set $Y = Y + 1$, Apakah $Y > X$?
Jika ya, ke Langkah 13.
Lainnya, kembali ke langkah 5.

Langkah 13: Tentukan Ms_{iter}, $MS_{iter} = \min \{MS_e\}$
Dan LK_{iter}.

Langkah 14: Perbaharui MS_{Th}.

Langkah 15: Set iter = iter + 1, Apakah iter > maxiter ?
Jika ya, ke Langkah 15.
Lainnya, ke Langkah 3.

Langkah 16: Tentukan MS_{best} dan LK_{best}
$MS_{best} = \min \{ MS_O, MS_{iter} \}$

Langkah 17: Tahap *local search* selesai.
Tampilkan *gantt chart* MS_{best}.

Sub Algoritma Pertukaran Dua Operasi Pada Lintasan Kritis.

Langkah 1: Tentukan dua operasi yang akan dipertukarkan (operasi *a* dan *b*) yang berada pada lintasan kritis dan berada pada sumber yang sama (mesin paralel *m* yang sama), dimulai dari dua operasi yang paling akhir.

Langkah 2: Tentukan himpunan operasi yang tidak perlu dijadwalkan kembali yaitu $\Omega_{PM[a]}$, $\Omega_{PJ[a]}$, $\Omega_{PJ[b]}$, dan $\Omega_{PJ[PM[a]]}$.

Langkah 3: Tentukan *ready time* dan *completion time b* dengan menggunakan persamaan sebagai berikut.

Untuk mesin paralel *m* sumber terjadinya pertukaran:

- Jika $\{ R_{PM[a]m} + t_{PM[a]m} \} > R_{PJ[b]} + t_{PJ[b]}$, maka
  tetapkan $R_{[b]m} = \{ R_{PM[a]m} + t_{PM[a]m} \}$
- Jika $\{ R_{PM[a]m} + t_{PM[a]m} \} \leq R_{PJ[b]} + t_{PJ[b]}$, maka
  tetapkan $R_{[b]m} = \{ R_{PJ[b]} + t_{PJ[b]} \}$

Untuk mesin paralel *m* lainnya:

- Jika $R_{km} > \{ R_{PJ[b]} + t_{PJ[b]} \}$, maka
  tetapkan $R_{[b]m} = R_{km}$
- Jika $R_{km} \leq \{ R_{PJ[b]} + t_{PJ[b]} \}$, maka
  tetapkan $R_{[b]m} = R_{PJ[b]} + t_{PJ[b]}$

Untuk menentukan *completion time b*, menggunakan persamaan sebagai berikut.

$C_{[b]} = \min\{R_{[b]m} + t_{ijkm}\}$

Jika memiliki nilai yang sama, pilih secara random.

Perbaharui nilai R_i dan R_{km} dengan persamaan sebagai berikut.

- $R_i = C_{[b]}$
- Untuk mesin paralel *m* terpilih, tetapkan $R_{km} = C_{[b]}$
- Untuk *m* lainnya, tetapkan $R_{km} = R_{km}$

Kemudian tentukan ready time dan completion time operasi a menggunakan

persamaan sebagai berikut.

- Jika { R_{km}> { $R_{PJ[a]}$ + $t_{PJ[a]}$, maka
  tetapkan $R_{[a]m} = R_{km}$
- Jika { $R_{km} \leq$ { $R_{PJ[a]}$ + $t_{PJ[a]}$, maka
  tetapkan $R_{[a]m} = R_{PJ[a]}$+ $t_{PJ[a](MWKR}$

Untuk menentukan *completion time a*, menggunakan persamaan sebagai berikut.

$C_{[a]} = \min\{R_{[a]m} + t_{ijkm}\}$

Perbaharui nilai R_i dan R_{km} dengan persamaan sebagai berikut.

- $R_i = C_{[a]}$
- Untuk mesin paralel *m* terpilih, tetapkan $R_{km} = C_{[a]}$
- Untuk *m* lainnya, tetapkan $R_{km} = R_{km}$

Langkah 4: Tentukan Ω_{RS} dengan ketentuan,
$\Omega_{RS} \notin \Omega_{PM[a]}$, $\Omega_{RS} \notin \Omega_{PJ[a]}$, $\Omega_{RS} \notin \Omega_{PJ[b]}$, dan $\Omega_{RS} \notin \Omega_{PJ[PM[a]]}$.
Tentukan *V* dan $f = 0$

Langkah 5: Tentukan *S*

Langkah 6: Jika $|S| > 1$, maka pilih *job* yang akan dijadwalkan terlebih dahulu menggunakan prioritas *Most Work Remaining* (MWKR). Jika job $\in S$ memiliki jumlah operasi yang sama maka pilih salah satu dari *S* secara *random*, dan lanjut ke langkah 7.
Lainnya ke Langkah 8.

Langkah 7: Pilih operasi O_{ijk} yang berada pada urutan paling awal *job* tersebut.
Tentukan *ready time* untuk kedua mesin dan *completion time* operasi O_{ijk} dengan menggunakan Persamaan sebagai berikut.

- Jika $R_{km} > R_i$, maka tetapkan $R_{ijkm} = R_{km}$
- Jika $R_{km} \leq R_i$, maka tetapkan $R_{ijkm} = R_i$

Untuk *Completion time*, $C_{ijk} = \min \{ R_{ijkm} + t_{ijkm} \}$
Perbaharui nilai R_{km} dan R_i dengan menggunakan persamaan berikut.
Perbaharui nilai R_i dan R_{km} dengan persamaan sebagai berikut.

- $R_i = C_{ijk}$
- Untuk mesin paralel *m* terpilih, tetapkan $R_{km} = C_{ijk}$
- Untuk *m* lainnya, tetapkan $R_{km} = R_{km}$

Langkah 8: Apakah semua operasi terjadwalkan, $f = V$?
Jika ya, ke Langkah 21.
Lainnya, kembali ke Langkah 19 dan keluarkan operasi O_{ijk} dari Ω_{RS},
$\Omega_{RS} = \Omega_{RS} - O_{ijk}$.

Langkah 9: Berhenti dan hitung makespan keseluruhan operasi, $MS_e = \max \{C_{ijk}\}$

4. PENGUJIAN ALGORITMA

Pengujian algoritma dilakukan dengan menggunakan dua skenario, yaitu Tahap 1 bertujuan untuk menguji cara kerja algoritma usulan, dan Tahap 2 bertujuan untuk menguji keandalan algoritma usulan dengan hasil dari penelitian-penelitian sebelumnya. Set data yang digunakan pada Tahap 1 dan Tahap 2 adalah set data dari penelitian Puryani (2003).

a. Tahap 1

Set data yang digunakan pada Tahap 1 dapat dilihat pada Tabel 1.

Tabel 1.Matriks Waktu Proses Kelompok Mesin Paralel Homogen (a) dan Heterogen (b) Kasus 1

Job	Operasi					
	1		2		3	
	m1	m2	m1	m2	m1	m2
A	m2	4	8	8	4	4
B	m3	5	6	6	2	2
C	m4	2	6	6	9	9
D	m5	4	9	9	2	2
E	m6	6	7	7	2	2

(a) Homogen

Job	Operasi					
	1		2		3	
	m1	m2	m1	m2	m1	m2
A	4	4	10	5	7	1
B	9	1	5	6	2	2
C	1	3	8	4	9	8
D	3	5	8	9	2	1
E	6	6	10	3	6	4

(b) Heterogen

b. Tahap 2

Set data yang digunakan pada Tahap 2 dapat dilihat pada Tabel 2 dan Tabel 3.

Tabel 2. Matriks Waktu Proses Kelompok Mesin Paralel Homogen (a) dan Heterogen (b) Kasus 2

Job	Operasi						
	1		2		3		
	m1	m2	m1	m2	m1	m2	m3
A	5	5	2	2	6	6	6
B	2	2	4	4	6	6	6
C	3	3	5	5	6	6	6
D	3	3	8	8	7	7	7
E	5	5	7	7	8	8	8

(a) Homogen

Job	Operasi						
	1		2		3		
	m1	m2	m1	m2	m1	m2	m3
A	8	4	2	2	8	6	4
B	1	2	1	6	4	9	3
C	6	3	3	7	10	6	3
D	3	2	9	6	9	8	4
E	7	2	4	9	9	6	8

(b) Heterogen

Tabel 3. Matriks Waktu Proses Kelompok Mesin Paralel Homogen (a) dan Heterogen (b) Kasus 3

Job	Operasi								
	1			2			3		
	m1	m2	m3	m1	m2	m3	m1	m2	m3
A	6	6	-	7	7	7	6	6	6
B	8	8	8	5	5	-	3	3	3
C	3	3	-	3	3	3	4	4	4
D	7	7	7	4	4	-	3	3	3
E	3	3	3	8	8	-	6	6	6

(a) Homogen

Job	Operasi								
	1			2			3		
	m1	m2	m3	m1	m2	m3	m1	m2	m3
A	1	10	-	1	10	9	2	6	9
B	7	5	10	6	3	-	4	3	2
C	2	3	-	1	5	2	1	8	2
D	2	9	10	3	5	-	1	1	7
E	4	3	2	6	10	-	10	7	1

(b) Heterogen

Hasil perbandingan makespan antara algoritma usulan dengan beberapa penelitian yang lain dapat dilihat pada Tabel 4 dan Tabel 5.

Tabel 4. Perbandingan *Makespan*Untuk Mesin Paralel Homogen

Kasus	Jumlah Job	SK			Makespan					
		1	2	3	Puryani (2003)	Luis et al. (2003)	Putra (2010)	Algoritma Usulan		
								10%	5%	15%
Kasus 1	5	2	2	2	23	22	22	22	22	22
Kasus 2	5	2	2	3	23	22	21	21	21	21
Kasus 3	5	2	3	3	19	19	19	19	19	19

Tabel 5. Perbandingan *Makespan*Untuk Mesin Paralel Heterogen

Kasus	Jumlah Job	SK			Makespan							
		1	2	3	Puryani (2003)	Luis et al. (2004)	Suryono (2004)	Saleh et al. (2005)	Putra (2010)	Algoritma Usulan		
										10%	5%	15%
Kasus 1	5	2	2	2	18	17	16	16	16	16	16	16
Kasus 2	5	2	2	3	17	17	17	17	17	17	17	17
Kasus 3	5	2	3	3	12	12	12	12	12	12	12	12

5. ANALISIS

Berdasarkan Tabel 4 dapat dilihat bahwa untuk kasus 1 mesin paralel homogen, algoritma GRASP with *Fixed Threshold* menghasilkan nilai yang lebih baik dari Puryani (2003) sebesar 1 satuan waktu, sedangkan pada penelitian Luis *et al.* (2003) dan Putra (2010) menghasilkan nilai makespan terbaik yang sama dengan algoritma usulan. Pada kasus 2 mesin paralel homogen, algoritma usulan lebih baik dari Puryani (2003) dan Luis *et al.* (2003), dan memiliki nilai makespan terbaik yang sama dengan penelitian Putra (2010). Untuk kasus 3 mesin paralel homogen, algoritma usulan mendapatkan makespan yang sama dengan penelitian-penelitian sebelumnya.

Berdasarkan Tabel 5dapat dilihatbahwauntuk kasus 1 mesin paralel heterogen. Algoritma usulan memiliki nilai makespan yang sama dengan Suryono (2004), Saleh *et al.* (2005), dan Putra (2010). Algoritma usulan mendapatkan makespan yang lebih baik dibandingkan penelitian Puryani (2003) dan Luis *et al.* (2004). Untuk kasus 2 dan kasus 3 mesin paralel heterogen, memberikan makespan yang sama dibandingkan penelitian-penelitian sebelumnya.

Apabila dilihat dari perbandingan-perbandingan tersebut, algoritma GRASP with *Fixed Threshold* memiliki keandalan dalam menyelesaikan penjadwalan *job shop* kelompok mesin paralel. Pada penelitian ini, persentase *threshold* yang digunakan adalah 10%. Selain menggunakan nilai *threshold* sebesar 10%, digunakan juga nilai *threshold* 5% dan 15%. Semua memberikan *makespan* terbaik yang sama untuk kedua set data. Perbedaannya terdapat pada tahap *local search*. Semakin kecil nilai *threshold* maka semakin sedikit kemungkinan solusi yang didapat.

6. KESIMPULAN

Kesimpulan dari hasil penelitian ini adalah:

1. Algoritma usulan yang dikembangkan adalah algoritma *greedy randomized adaptive search with fixed threshold* yang diujikan dalam menyelesaikan masalah penjadwalan *job shop* kelompok mesin paralel dengan kriteria minimisasi *makespan*.

2. Pengujian tahap 1 dan tahap 2 menunjukkan bahwa algoritma usulan dapat digunakan untuk menyelesaikan masalah penjadwalan *job shop* kelompok mesin paralel dan makespan yang dihasilkan sama dengan penelitian sebelumnya.

REFERENSI

Adams J., Balas, E., dan Zawack, D. (1988). *The Shifting Bottleneck Procedure for Job Shop Scheduling*, *Management Science*, Vol. 34 (3) page 391-401.

Brucker, P., Jurisch, B., dan Sievers, B. (1994). *A Branch and Bound Algorithm for Job Shop Scheduling Problem*, *Discrete Applied Mathematics*. Vol. 49 page 105-127.

Carlier, J., Dan Pinson, E. (1989). *An Algorithm for Solving the Job Shop Problem*, *Management Science*. Vol. 35 (2) page 146-176.

Dueck, G., Dan Scheuer, T. (1990). *Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing*. Journal Of Computational Physics. Vol. 90 (1) page 161-175.

Garey, M. R. dan Johnson, D. S. (1979). *Computers and Intractability*: *A Guide to the Theory of NP Completeness*. W. H. Freeman and Co. New York.

Puryani. (2003). *Penjadwalan Job Shop Kelompok Mesin Paralel Menggunakan Algoritma Jadwal Aktif dan Algoritma Branch And Bound Like (BABL) Untuk Kriteria Minimasi Makespan*. Tugas Sarjana – Program Studi Teknik Industri, Institut Teknologi Nasional Bandung.

Putra, H. P. (2010). *Algoritma Penjadwalan Kelompok Mesin Paralel Homogen dan Heterogen Menggunakan Greedy Randomized Adaptive Search Procedure untuk Kriteria Minimasi Makespan.*Tugas Sarjana – Program Studi Teknik Industri, Institut Teknologi Nasional Bandung.

Scaparra, M.P., Church, R.L. (2005). *A GRASP and Path Relinking Heuristic for Rural Road Network Development*, *Journal of Heuristics*. Vol. 11 (1) page 89-108.

Sule, D. R. Dan Vijayasundaram, K. (1998). *A Heuristic Procedure for Makespan Minimization In Job Shop with Multiple Identical Processors*, *Computers and Industrial Engineering*. Vol. 35 page 399-402.