ALGORITMA PENJADWALAN *NO-WAIT JOB SHOP* MENGGUNAKAN *GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE WITH FIXED THRESHOLD* DENGAN KRITERIA MINIMISASI *MAKESPAN*[*]

KENNO PRASETYO, EMSOSFI ZAINI, ARIF IMRAN

Jurusan Teknik Industri
Institut Teknologi Nasional (Itenas) Bandung

Email: Kennoprasetyo@yahoo.com

ABSTRAK

Pada kenyataannya, beberapa perusahaan industri tidak mengijinkan adanya kondisi no-wait. Kondisi no-wait, yaitu kondisi yang tidak memperkenankan adanya delay, dimana saat selesai operasi dari suatu job di suatu mesin harus sama dengan saat mulai operasi dari job tersebut pada mesin berikutnya. Penelitian ini membahas tentang metode metaheuristik untuk menyelesaikan masalah no-wait job shop. Algoritma yang digunakan yaitu greedy randomized adaptive search procedure (GRASP) with fixed threshold dengan kriteria minimasi makespan. Algoritma ini mempunyai dua tahap. Tahap pertama merupakan tahap konstruksi yang menghasilkan solusi inisial. Tahap kedua merupakan tahap local search untuk memperbaiki jadwal inisial dengan melakukan proses insert dan exchange. Nilai fixed threshold ditambahkan untuk membuka batas makespan yang diijinkan, sehingga kemungkinan untuk mendapatkan makespan dengan nilai lebih kecil akan lebih besar. Algoritma usulan diuji menggunakan set data dari literatur. Hasil yang didapat menunjukkan hasil yang sama baiknya dengan penelitian yang sebelumnya.

***Kata kunci**: Penjadwalan, No-Wait Job Shop, GRASP, Fixed Threshold*

ABSTRACT

In the fact, some industrial companies doesn't allow no-wait condition. it's a condition that doesn't allow of any delay, which completed the operation from a job in a machine should be the same with the start of operations of the job on the next machine. This research discusses about metaheuristic method to resolve no-wait job shop problem. It's used greedy randomized adaptive search procedure (GRASP) with fixed threshold with mekespan minimization criteria. It's has two stages. First stage is construction stage which generates initial solution. Second stage is local search stage to fixing initial solution by doing insert and exchange process. Fixed threshold value added to open makespan permitted, so that a chances of getting an makespan with a smaller value will be greater.

[*] *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.*

Porposed algorithm is tested using data sets form literature. The result obtained showed the same result as good as previous research.

***Keywords**: Scheduling, No-Wait Job Shop, GRASP, Fixed Threshold*

1. PENDAHULUAN

1.1 Pengantar

Permasalahan penjadwalan *job shop* merupakan definisi dari penjadwalan n *job* ($J_i = J_1, J_2,.., J_n$) pada *m* mesin (k = 1, 2,.., m), dan setiap *job* memiliki *routing* ($O_{i1}, O_{i2},.., O_{ij}$) masing-masing yang unik, dimana tidak perkenankan adanya gangguan pada mesin-mesin yang telah ditentukan dalam selang waktu operasi tertentu pada setiap operasinya. Tujuan dari penjadwalan yaitu untuk menghasilkan jadwal yang dapat meminimumkan waktu yang dibutuhkan untuk memproses seluruh *job* pada semua mesin (*makespan*).

Permasalahan penjadwalan *job shop* dapat diselesaikan dengan menggunakan beberapa metode diantaranya teknik optimasi, metode heuristik, dan metaheuristik. Teknik optimasi digunakan untuk menyelesaikan kasus-kasus kecil dengan waktu yang relatif pendek dapat menghasilkan solusi yang optimal. Oleh karena itu, metode-metode berbasis heuristik dikembangkan untuk menyelesaikan permasalahan *job shop* yang kompleks. Metode ini tidak menjamin menghasilkan solusi optimal, tetapi dapat menghasilkan solusi dalam waktu yang relatif singkat. Selanjutnya dikembangkan pendekatan modern heuristik (metaheuristik) untuk menyelesaikan masalah *job shop*. Metode metaheuristik merupakan pengembangan dari metode heuristik. Namun dalam beberapa permasalahan *job shop* masih terdapat kondisi *delay* diantara dua mesin atau dua operasi pada *job* yang sama.

Pada kenyataannya terdapat beberapa industri yang tidak mengijinkan adanya *no-wait* seperti industri baja, plastik, kimia, dan makanan. Kondisi *no-wait*, yaitu kondisi yang tidak memperkenankan adanya delay, dimana saat selesai operasi dari suatu job di suatu mesin harus sama dengan saat mulai operasi dari job tersebut pada mesin berikutnya. Memperhatikan uraian tersebut, belum terlihat adanya algoritma *GRASP with fixed thresold* dengan kriteria minimasi *makespan* untuk menyelesaikan masalah *no-wait job shop.*

1.2 Identifikasi Masalah

Terdapat beberapa industri yang tidak mengijinkan adanya *no-wait* seperti industri baja, plastik, kimia, dan makanan. *No-wait job shop* meruupakan penugasan sejumlah *job* pada sejumlah mesin dengan setiap *job* memiliki urutan proses yang unik dan setiap operasi dari *job* yang sama harus diproses tanpa adanya interupsi. Metode pemecahan masalah yang digunakan pada penelitian ini adalah algoritma *GRASP with fixed thresold* dengan kriteria minimasi *makespan.*

2. STUDI LITERATUR

2.1 Konsep Dasar Penjadwalan

Baker (1974) penjadwalan yaitu proses pengalokasian sumber-sumber untuk meimilih sekumpulan tugas dalam jangka waktu tertentu. Definisi ini memiliki dua arti, yaitu:

1. Penjadwalan merupakan fungsi pengambilan keputusan.
2. Penjadwalan merupakan suatu teori.

2.2 Penjadwalan *Job Shop*

Ciri khas persoalan *job shop* adalah aliran pekerjaan dalam *job shop* tidak searah (*non*

unidirectional), karena aliran proses dari setiap *job* yang tidak searah, maka mengakibatkan setiap *job* dapat diproses lebih dari satu kali pada mesin yang sama. *Job* yang diproses dapat berupa *job* baru atau *job* yang sudah dikerjakan (*work in process*).

2.3 Penjadwalan *No-Wait Job Shop*

Penjadwalan *no-wait job shop* (NJWS) merupakan penjadwalan yang tidak mengijinkan adanya waktu tunggu antara dua operasi berurutan pada suatu *job*, sehingga saat selesai operasi dari suatu *job*.

2.4 Heuristik Dan Metaheuristik

Reeves dan Beasley (1993) mendefinisikan heuristik sebagai suatu teknik yang digunakan untuk mencari solusi yang baik (mendekati solusi yang optimal) dengan menggunakan perhitungan yang layak dan solusi yang dihasilkan tidak dapat dijamin kelayakan dan keoptimalannya atau bahkan dapat digunakan untuk menyatakan seberapa dekat keoptimalan tertentu dari solusi yang mungkin dihasilkan.

Menurut Osman dan Kelly (1996) metaheuristik adalah proses pengulangan generasi yang memandu heuristik sebelumnya dengan mengkombinasikan konsep cerdas yang berbeda untuk menyelidiki dan memanfaatkan ruang pencarian dengan menggunakan strategi belajar untuk mendapatkan struktur informasi dalam rangka secara efisien menemukan solusi yang mendekati optimal.

2.6 *Greedy Randomized Adaptive Search Procedure* (GRASP)

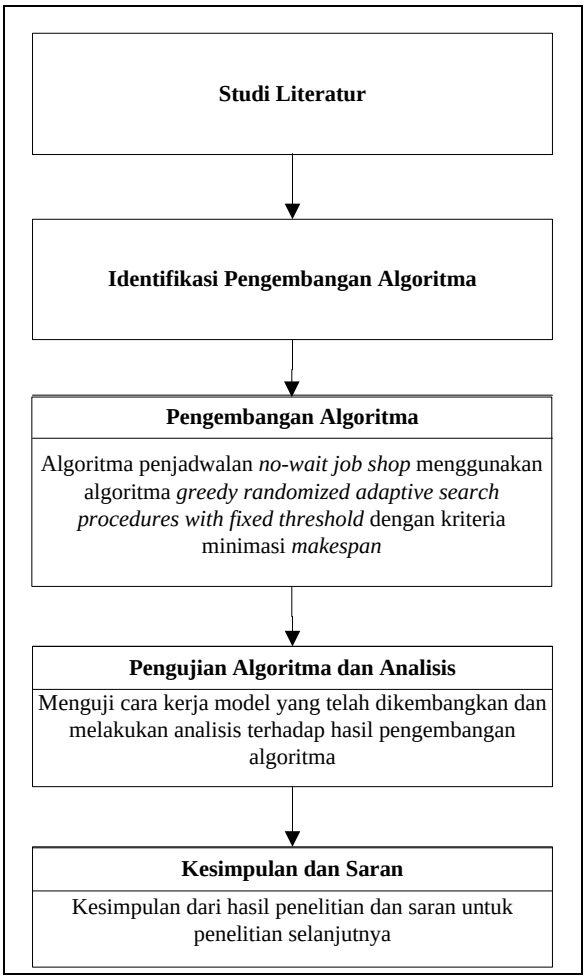
Greedy Randomized Adaptive Search Procedure (GRASP) adalah sebuah metode metaheuristik yang didasarkan pada teknik pedekatan *multi-start randomized* untuk menyelesaikan permasalahan optimasi kombinatorial (*NP-hard*). GRASP memiliki dua tahap, yaitu tahap konstruksi dan *local search*. Pada tahap konstruksi, *Restricted Candidate List* (RCL) dibentuk untuk mendapatkan solusi inisial. Sementara pada tahap *local search*, solusi inisial yang diperoleh dari tahap konstruksi diperbaiki.

2.7 *Threshold*

Algoritma 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 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. Nilai *threshold* yang digunakan secara berangsur-angsur akan meningkat selama proses pencarian dan akan mendekati nol setelah akhir pencarian. Hal tersebut menandakan hanya perpindahan yang lebih baik yang diterima. Selain *threshold* terdapat juga *fixed threshold* dimana nilai *threshold* yang digunakan pada setiap iterasinya sama atau tidak berubah-ubah.

3. METODOLOGI PENELITIAN

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



Gambar 1. Tahapan Penelitian

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

		MAKESPAN					
		Optimasi	Heuristik	Metaheuristik			
				Genetic Algorithm	CLLM	GRASP	GRASP With *Fixed Threshold*
Job Shop	Standar	Carlier dan Pinson (1998) Brucker *et al. (1994)*	Adams et al. (1998) Wenqi & Aihua (2004) Chu *et al. (1998)*	Zhou *et al*. (2001) Watanabe *et al.*		Binato *et al. (2001)*	Usman (2014)
	No Wait	Mascis dan Pacciarelli (2002)	Bansal *et al. (2005)*		Zhu *et al.* (2008)	Kamilah (2010)	PENELITIAN

Gambar 2. Peta Posisi Penelitian

4. PENGEMBANGAN ALGORITMA

Notasi-notasi yang digunakan dalam algoritma penelitian adalah:
J_i : *Job* ke-*i* (*i* = 1,2,...,*n*)
$\sum_{j=1}^{k} p_{ij}$: Total waktu proses untuk setiap *job* ke-*i*
t_{ij} : Waktu proses operasi ke-*j job* ke-*i*
R_i : *Ready time job* ke-*i*
R_k : *Ready time* mesin ke-*k*
L_{min} : Jumlah *job* minimum yang terdapat dalam RCL
B : Jumlah makasimum lintasan kritis yang dipertimbangkan

C_i : *Completion time* untuk setiap *job* ke-*i*
H : Pasangan *job*
h(J_i) : Fungsi *greedy* (nilai makespan dari penjadwalan J_i pada *job* yang telah terjadwalkan)
$\underline{h}$: Nilai C_i minimum
$\overline{h}$: Nilai C_i maksimum
MS_x : *Makespan*
MS_{Th} : Batas atas nilai *makespan* berdasarkan *%Th*
MS_{iter} : *Makespan* iterasi ke *iter*
MI : *Makespan* jadwal inisial
M : *Makespan* terpilih

Algoritma GRASP *with fixed threshold* dengan kriteria minimasi *makespan* dapat dilihat sebagai berikut:

Tahap I : Tahap Konstruksi

Langkah 1
Input : - Jumlah *job* (n)
- Matriks *routing*
- Matriks waktu proses

Langkah 2
Set : - R_i = 0, $\forall_i$
- R_k = 0, $\forall_k$
- L_{min} = maks (2, $[^n/_3]$)
- *Maxiter* = maks (2, $[^{2n}/_5]$)

Langkah 3
Set iterasi = 1

Langkah 4
Tentukan himpunan *job* yang siap dijadwalkan (Ω)

Langkah 5
Set r = 1 (r = 1,2,...,Ω)

Langkah 6
Hitung C_i dengan persamaan :
$C_i = R_i + \sum_{j=1}^{k} p_{ij}, \forall i \in \{\Omega\}$ {1}

Langkah 7
Tentukan : $\underline{h}$ = min $\{C_i\}$
$\overline{h}$ = maks $\{C_i\}$

Langkah 8
Tentukan secara random nilai parameter α, dengan $\alpha \in [0,1]$
Bentuk *Restricted Candidate List* (RCL) dengan menggunakan persamaan :
RCL = $\{J_i \in \Omega \,|\underline{h} \leq h(J_i) \leq \underline{h} + \alpha(\overline{h} - \underline{h})\}$ {2}

Langkah 9
Periksa apakah |RCL| $\geq L_{min}$?
Jika ya, lanjutkan ke langkah 10
Lainnya, urutkan *job-job* dalam Ω berdasarkan C_i terkecil ke terbesar dan ambil *job* sebanyak L_{min} untuk masuk ke dalam RCL dan lanjutkan ke langkah 10

Langkah 10
Pilih satu *job* dalam RCL secara *random*, J^* dan jadwalkan dengan cara saat mulai operasi pertama $J_{[i]}$ sama dengan saat selesai operasi terakhir $J_{[i-1]}$

Langkah 11
Hilangkan *job* J^* dari Ω, $\Omega = \Omega - \{J^*\}$ dan perbaharui R_k, $\forall k$

Langkah 12
Periksa apakah $|\Omega| = 1$?
Jika ya, maka jadwalkan *job* dalam Ω dan lanjutkan ke langkah 13
Lainnya, set $r = r + 1$ dan kembali ke langkah 6
Langkah 13
Hitung *makespan* (*MI*) dengan menggunakan persamaan: $MI = \text{maks}\{C_i\}$
Dan tampilkan *gantt chart* dari jadwal yang dihasilkan

Tahap II : Tahap *Local Search*
Langkah 14
Input : Jadwal inisial dari langkah 13
Langkah 15
Set $B = [{}^{n}/_{2}] + 1$
Langkah 16
A) Secara berurutan setiap jadwal yang terbentuk, geser kiri (*left shift*) semua *job* dengan cara menggeser job tersebut sehingga saat mulai operasi pertama $J_{[i]}$ sama dengan saat selesai salah satu operasi $J_{[i-1]}$
B) Hitung *makespan* setelah proses geser kiri
Langkah 17
Lakukan proses *insert* dengan cara memeriksa selang waktu kosong dari setiap mesin, kemudian sisipkan job dan hitung *makespan*-nya
Lakukan langkah ini sampai memperoleh *makespan* terbaik. Nyatakan *makespan* sebagai M_x.
Langkah 18
Hitunglah nilai $MS_{Th} = M_x + (M_x \text{ x \% } Th)$ {3}
Langkah 19
Tentukan lintasan kritis dan pilih sebanyak *B* lintasan untuk dipertimbangkan
Langkah 20
Tentukan himpunan pasangan-pasangan job yang akan dipertukarkan (*H*) pada lintasan kritis
Langkah 21
A) Ambil salah satu pasangan dalam *H*, kemudian lakukan pertukaran *job* dan tampilkan *Gantt chart* nya.
B) Periksa apakah urutan pada langkah 20 (A) sudah pernah terjadi ?
Jika ya, lanjutkan ke Langkah 24
Lainnya, lanjutkan ke Langkah 22
Langkah 22
Lakukan Langkah 17
Langkah 23
Periksa apakah $M_{x+1} \leq MS_{Th}$
Jika ya, Hitunglah nilai MS_{Th} dan kembali ke Langkah 19
Lainnya, lanjutkan ke langkah 24
Langkah 24
Periksa apakah H = { }?
Jika ya, periksa apakah *H* pada iterasi $x = \{\ \}$?
Jika ya, set M_{x+1} terkecil sebagai MS_{iter} dan tampilkan *Gantt chart*, lanjutkan ke Langkah 24
Lainnya, kembali ke langkah 21
Lainnya, kembali ke langkah 21

Langkah 25
Set *iter* = *iter* + 1

Periksa apakah *iter* > *maxiter* ?
Jika ya, selesai dan pilih jadwal dengan *makespan* terbaik:
M = min ($MS_1, MS_2, .., MS_{iter}$)
Tampilkan *gantt chart* hasil penjadwalannya
Lainnya, kembali ke langkah 5

5. PENGUJIAN ALGORITMA DAN ANALISIS

Pengujian algoritma dilakukan dengan menggunakan set data Framinan dan Schuster (2006), Zhu *et al*. (2008), Pan dan Huang (2009), serta Fisher dan Thompson (1963). Set data dapat dilihat pada Gambar 3 dan Gambar 4.

JOB	OPERASI		
	1	2	3
1	M2	M1	M3
2	M1	M3	M2
3	M3	M1	M2

Data Matriks *Routing*

JOB	OPERASI		
	1	2	3
1	2	4	1
2	3	4	5
3	1	6	1

Data Matriks Waktu Proses

Gambar 3. Set Data Framinan dan Schuster (2006)

JOB	OPERASI				
	1	2	3	4	5
1	M4	M2	M3	M5	M1
2	M5	M4	M2	M3	M1
3	M1	M2	M3	M4	M5

Data Matriks *Routing*

JOB	OPERASI				
	1	2	3	4	5
1	5	6	4	1	2
2	8	8	7	5	4
3	1	6	5	7	4

Data Matriks Waktu Prose

Gambar 4. Set Data Zhu *et al*. (2008)

JOB	OPERASI			
	1	2	3	4
1	M1	M4	M3	M2
2	M1	M3	M4	M2
3	M2	M4	M3	M1
4	M4	M2	M1	M3
5	M3	M2	M4	M1
6	M3	M4	M2	M1

Data Matriks *Routing*

JOB	OPERASI			
	1	2	3	4
1	7	10	7	4
2	10	8	5	7
3	3	2	1	3
4	8	1	8	1
5	3	4	7	3
6	1	6	6	10

Data Matriks Waktu Proses

Gambar 5. Set Data Pan dan Huang (2009)

JOB	OPERASI					
	1	2	3	4	5	6
1	M3	M1	M2	M4	M6	M5
2	M2	M3	M5	M6	M1	M4
3	M3	M4	M6	M1	M2	M5
4	M2	M1	M3	M4	M5	M6
5	M3	M2	M5	M6	M1	M4
6	M2	M4	M6	M1	M5	M3

Data Matriks *Routing*

JOB	OPERASI					
	1	2	3	4	5	6
1	1	3	6	7	3	6
2	8	5	10	10	10	4
3	5	4	8	9	1	7
4	5	5	5	3	8	9
5	9	3	5	4	3	1
6	3	3	9	10	4	1

Data Matriks Waktu Proses

Gambar 6. Set Data Fisher dan Thompson (1963)

Set data yang digunakan untuk contoh peritungan mengggunakan set data Framinan dan Schuster (2006). Langkah-langkah penyelesaian permasalahan *no-wait job shop* menggunakan algoritma GRASP *with fixed threshold* dengan kriteria minimisasi *makespan* adalah:

Tahap I : Tahap Konstruksi

Langkah 1

Input : - Jumlah *job* (n) = 3
- Matriks *routing* (lihat Gambar 5.1)
- Matriks waktu proses (lihat Gambar 5.2)

Langkah 2

Set : - R_i = 0, $\forall_i$
- R_k = 0, $\forall_k$
- L_{min} = maks (2, $[^3/_3]$) = 2
- *Maxiter* = maks (2, $[^{2x3}/_5]$) = 2

Langkah 3

Set *iter* = 1

Langkah 4

Himpunan *job* yang siap dijadwalkan (Ω) adalah

$\Omega = \{J_1, J_2, J_3\}$

Langkah 5

Set r = 1

Langkah 6

Hitung C_i dengan persamaan {1}

$C_1 = 0 + 7 = 7$

$C_2 = 0 + 12 = 12$

$C_3 = 0 + 8 = 8$

Langkah 7

Tentukan: $\underline{h}$ dan $\overline{h}$

$\overline{h}$ = maks {7,12,8} = 12

$\underline{h}$ = min {7,12,8} = 7

Langkah 8

Tentukan secara random nilai parameter α, dengan $\alpha \in [0,1] = 0{,}5$

Bentuk *Restricted Candidate List* (RCL) dengan menggunakan persamaan {2}

RCL = $\{7 \leq h(J_i) \leq 7 + 0{,}5\ (12 - 7)\}$

RCL = $\{7 \leq h(J_i) \leq 9{,}5\}$

Sehingga RCL yang terbentuk adalah RCL = $\{J_1, J_3\}$

Langkah 9

Periksa apakah |RCL| $\geq L_{min}$?

$2 \geq 2$?

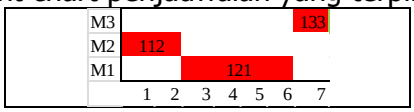
Ya, lanjutkan ke langkah 10

Langkah 10

RCL = $\{J_1, J_3\}$

Dipilih secara *random* elemen J^* dalam RCL adalah J_1.

Gambar 7. memperlihatkan *gant chart* penjadwalan yang terpilih dalam RCL.



Gambar 7. *Gantt Chart* Penjadwalan *Job* yang Terpilih Dalam RCL

Langkah 11

Hilangkan J^* dari Ω

$\Omega = \{J_1, J_2, J_3\} - \{J^*\}$

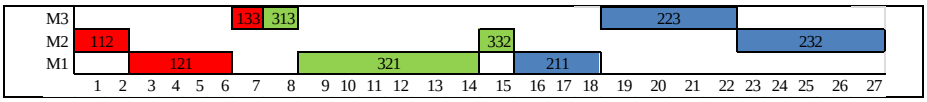
$\Omega = \{J_2, J_3\}$

dan perbaharui R_k, $\forall_k$

$R_1 = 6$; $R_2 = 2$; $R_3 = 7$

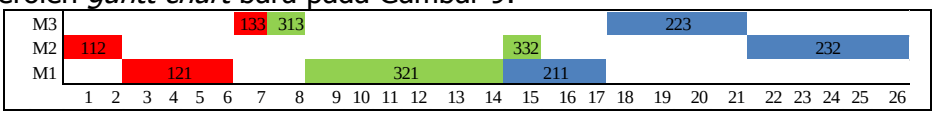
Langkah 12
Periksa apakah $|\Omega| = 1$?
Tidak, $|\Omega| = 2$, maka set $r = 1 + 1 = 2$ dan kembali ke langkah 6

Lakukan hal sama, hingga mendapatkan *Gantt chart* tahap konstruksi
Gantt chart hasil tahap konstruksi dapat dilihat pada Gambar 8. dan urutan jadwal yang terbentuk adalah J_1 - J_2 - J_3.



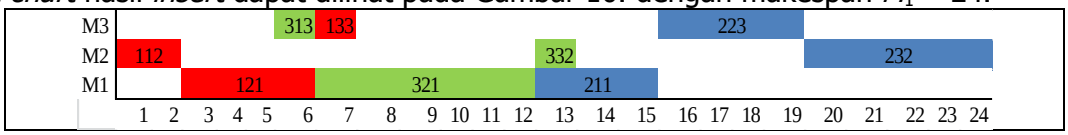
Gambar 8. *Gantt Chart* Hasil Tahap Konstruksi

Tahap II : Tahap *Local Search*
Langkah 14
Input : Jadwal inisial dari langkah 13
Langkah 15
Set $B = [^3/_2] + 1 = 3$
Langkah 16
A) Lakukan geser kiri (*left shift*) dari jadwal konstruksi pada Gambar 8. Sehingga diperoleh *gantt chart* baru pada Gambar 9.



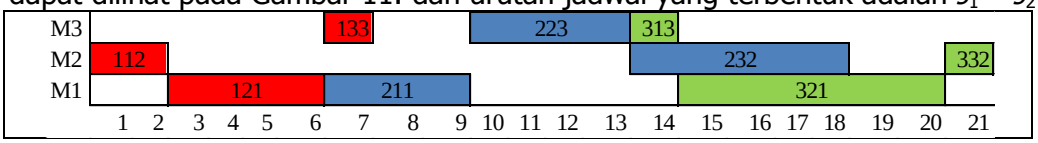
Gambar 9. *Gantt Chart* Hasil Geser Kiri

B) Hitung *makespan* setelah proses geser kiri. *Makespan* = 26
Langkah 17
Berdasarkan Gambar 9. dapat dilakukan proses *insert* dengan menyisipkan J_3 diantara J_2. *Gantt chart* hasil *insert* dapat dilihat pada Gambar 10. dengan makespan $M_1 = 24$.



Gambar 10. *Gantt Chart* Hasil *Insert*

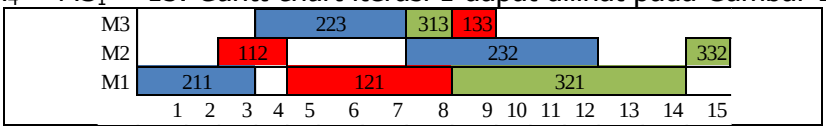
Langkah 18
Hitunglah nilai MS_{Th} dengan persamaan {3}
MS_{Th} = 24 + (24 x 10%) = 26
Langkah 19
Lintasan kritis pada Gambar 5.7 adalah:
112-121-321-211-223-232
Langkah 20
Himpunan pasangan-pasangan *job* yang akan dipertukarkan pada lintasan kritis adalah: $H = \{J_3 - J_2, J_1 - J_3\}$
Langkah 21
A) Pasangan *job* yang diambil dalam H adalah $J_3 - J_2$. Hasil pertukaran J_3 dengan J_2 dapat dilihat pada Gambar 11. dan urutan jadwal yang terbentuk adalah $J_1 - J_2 - J_3$.



Gambar 11. *Gantt Chartt* Hasil Pertukaran J_3 dengan J_2

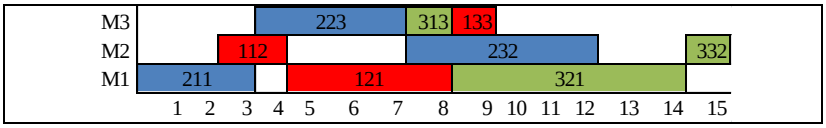
B) Urutan $J_1 - J_2 - J_3$ tidak pernah terjadi, maka lanjutkan ke Langkah 22.

Langkah 22
Berdasarkan Gambar 11. tidak ada *job* yang dapat dilakukan proses *insert* karena tidak ada waktu kosong diantara $J_1 - J_2$ dan *makespan* yang dihasilkan adalah M_2 = 21.
Langkah 23
Apakah $M_2 \leq MS_{Th}$?
21 $\leq$ 26?
Ya, maka nilai MS_{Th} = 21 + (21 x 10%) = 23, dan kembali ke langkah 19
Lakukan hal sama, hingga langkah 24
Langkah 24
Apakah H = { }?
Ya, maka set $M_4 = MS_1$ = 15. *Gantt chart* iterasi 1 dapat dilihat pada Gambar 12.



Gambar 12. *Gantt Chart* Hasil Iterasi 1

Langkah 25
Set *iter* = 1 + 1 = 2
Apakah iter (2) > *maxiter* (2)?
Tidak, maka kembali ke Langkah 5.
Lakukan hal sama, hingga mendapatkan *Gantt chart* terbaik. *Gantt chart* dengan *makespan* terbaik dilihat pada Gambar 13.



Gambar 13. *Gantt Chart* dengan *Makespan* Terbaik

Hasil *makespan* dengan menggunakan keempat set data dapat dilihat pada Tabel 1.

Tabel 1. Hasil *Makespan* Menggunakan 4 Set Data

Set Data	n	m	Framinan dan Schuster (2006)	Zhu et al. (2008)	Pan dan Huang (2009)	Kamilah (2010)	Model Usulan
Framinan dan Schuster (2006)	3	3	16			15	15
Zhu et al. (2008)	3	5		33		33	33
Pan dan Huang (2009)	6	4			56	49	49
Fisher dan Thompson (1963)	6	6	73	73	73	73	73

Berdasarkan Tabel 1. dapat dilihat bahwa algoritma GRASP *with fixed threshold* memberikan hasil yang cukup baik dalam menyelesaikan permasalahan penjadwalan *no-wait job shop*. Algoritma ini menghasilkan *makespan* yang sama dengan set data pada penelitian Kamilah (2010). Algoritma GRASP *with fixed threshold* menghasilkan *makespan* lebih baik sebesar 1 satuan waktu dibandingkaan set data Framinan dan Schuster (2006), serta lebih baik sebesar 7 satuan waktu dibandingkaan set data Pan dan Huang (2009). Berdasarkan hasil skenario 2, maka algoritma usulan memiliki keandalan untuk menyelesaikan permasalahan *no-wait job shop*.

6.1 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 *no-wait job shop* dengan kriteria minimisasi *makespan*.
2. Hasil yang diperoleh dari algoritma usulan sama baiknya dengan penelitian

sebelumnya yang dilakukan oleh Kamilah (2010).

3. Pengujian skenario 1 dan skenario 2 menunjukkan bahwa algoritma usulan dapat digunakan untuk menyelesaikan masalah penjadwalan *no-wait job shop*.
4. Penelitian selanjutnya dapat menggunakan *software* dalam membuat dan menguji algoritma ini, sehingga jumlah data yang akan diolah dapat lebih banyak.

REFERENSI

Baker, K. R., 1974, *Introducing To Squencingand Scheduling*, John Wiley and Sons, New York.

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

Fisher, H. dan Thompson, G. L., 1963. Probabilistic learning combinations of local job-shop scheduling rules, *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ.

Framinan, J.M. dan Schuster, C. J., 2006. An enhanced timetabling procedure for the no-wait job shop problem: a complete local search approach, *Computers and Operations Research*, 331, 1200-1213.

Kamilah, N., 2010, *Model Penjadwalan No-Wait Job Shop Menggunakan Algoritma Greedy Randomized Adaptive Search Procedures Dengan Kriteria Minimasi Makespan*. Tugas Sarjana – Program Studi Teknik Industri, Institut Teknologi Nasional, Bandung.

Osman, I.H., dan Kelly, J.P., 1996. Meta-heuristics: An Overview In Osman, I.H., Kelly, J.P., editors, *Meta-heuristics: Theory and Application*. Kluwer Academic Publishers, Massachusetts, 1-21.

Pan, Jason Chao-Hsien., dan Huang, Han-Chiang., 2009. A hybrid genetic algorithm for no-wait job shop scheduling problems. *Expert Systems with Application*, 36: 5800–5806.

Reeves, C.R., Dan Beasly, J.E., 1993, *Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications*, Oxford.

Zhu, J., Li, X., dan Wang, Q., 2008. Complete local search with limited memory algorithm for no-wait job shops to minimize makespan, *European Journal of Operational Research*, doi : 10.1016/j.ejor.2008.09.015.