Algoritma Penjadwalan *Job Shop* Alternatif *Routing* Menggunakan *Variable Neighborhood Descent With Fixed Threshold* Untuk Minimisasi *Makespan*[*]

SEPTIANI HARTINI, EMSOSFI ZAINI, ARIF IMRAN

Jurusan Teknik Industri
Institut Teknologi Nasional (Itenas) Bandung

Email: septi.tetep@gmail.com

ABSTRAK

Makalah ini membahas algoritma penjadwalan job shop alternatif routing menggunakan variable neighborhood descent (VND) with fixed threshold dengan kriteria minimisasi makespan. Tahap-tahap yang dilakukan pada algoritma ini yaitu tahap konstruksi dan tahap local search. Pada tahap konstruksi, urutan jadwal awal dibuat menggunakan penjadwalan non delay yang telah dimodifikasi. Pada tahap local search perbaikan urutan jadwal menggunakan proses insert dan exchange terhadap struktur neighborhood secara deterministik. Data-data hipotetik yang digunakan pada makalah ini merupakan data-data dari literatur.

***Kata kunci**: Job shop, alternatif routing, variable neighborhood descent with fixed threshold*

ABSTRACT

This paper discusses the scheduling algorithm of job shop with alternative routing using variable neighborhood descent (VND) with a fixed threshold for the makespan minimization criteria. The stages are performed on this algorithm, namely the construction phase and stage of local search. On the construction phase the sequence of initial schedules created using the non delay scheduling that has been modified. At this stage of local search repair sequence schedule by using the insert and exchange process of deterministic structure in neighborhood. Hipotetik Data used in this paper is the data from the literature.

***Keywords**: Job shop, alternative routing, variable neighborhood descent with fixed threshold*

[*] *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

1.1 Pengantar

Masalah penjadwalan muncul ketika terdapat berbagai macam tugas (*job*) atau proses yang harus dilakukan, sedangkan sumber daya (waktu, bahan baku, tenaga kerja, mesin, modal, dan sebagainya) yang dibutuhkan untuk menyelesaikan tugas-tugas atau proses tersebut terbatas sehingga diperlukan suatu pengaturan atas pelaksanaan tugas-tugas atau proses-proses tersebut.

Seiring perkembangan teknologi, sebuah mesin tidak hanya dapat melakukan satu jenis proses operasi saja. Misalnya, mesin *milling* dapat digunakan untuk melakukan proses yang sama dengan mesin *drilling* yaitu membuat lubang pada semua material dengan mengganti mata pahat. Kemampuan teknologi mesin mengakibatkan sebuah operasi *job* dapat dikerjakan di beberapa mesin. Sehingga sebuah operasi *job* dapat memiliki beberapa alternatif mesin yang disebut juga dengan alternatif *routing*. Nasr dan Elsayed (1990) membahas permasalahan *job shop* alternatif mesin menggunakan *greedy procedure* untuk meminimisasi *mean flow time*. Penelitian ini hanya menghasilkan satu solusi karena algoritma yang digunakan hanya menghasilkan sebuah solusi.

Variable Neighborhood Descent (VND) merupakan metode metaheuristik yang dapat menyelesaikan masalah-masalah optimasi solusi kombinatorial dengan cara perubahan *neighborhood* yang berbeda-beda secara terstruktur (Mladenovic dan Hansen, 1997). Pada VND, perubahan *neighborhood* dilakukan secara deterministik. Sevkali dan Aydin (2006) menggunakan *variable neighborhood search* untuk menyelesaikan permasalahan *job shop* dengan kriteria minimisasi *makespan*. Benny (2009) telah menyelesaikan model penjadwalan *job shop* alternatif *routing* dengan menggunakan algoritma VND dengan kriteria minimisasi *makespan.*

Threshold Accepting dikembangkan oleh Dueck dan Scheuer (1990). Sama dengan VND, TA 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. VND dan *fixed threshold* belum pernah digunakan secara bersamaan untuk menyelesaikan algoritma penjadwalan *job shop* alternatif *routing.* Algoritma *fixed threshold* memungkinkan solusi tidak terjebak pada minimum lokal dengan cara menerima solusi baru yang mengarah ke nilai yang lebih tinggi.

1.2 Identifikasi Masalah

Pada kondisi nyata, sebuah mesin dapat memiliki beberapa fungsi teknologi yang dapat melakukan berbagai proses operasi. Kemampuan teknologi mesin ini yang menyebabkan sebuah operasi *job* dapat dikerjakan pada beberapa mesin yang disebut dengan alternatif mesin atau alternatif *routing.* Untuk itu diperlukan pengembangan suatu algoritma penjadwalan *job shop* dengan alternatif *routing* menggunakan *variable neighborhood descent with fixed threshold* dengan kriteria minimasi *makespan.*

2. STUDI LITERATUR

2.1 Konsep Dasar Penjadwalan

Persoalan penjadwalan timbul apabila beberapa pekerjaan akan dikerjakan secara bersamaan, sedangkan sumber yang dimiliki terbatas. *Input* dari suatu penjadwalan mencakup jenis dan banyaknya *part* yang akan dioperasi, urutan ketergantungan antar operasi, waktu proses untuk masing-masing operasi, serta fasilitas yang dibutuhkan oleh

setiap operasi. Sedangkan *output* dari penjadwalan meliputi *dispatch list*, yaitu daftar yang menyatakan urutan pemrosesan *part* serta waktu mulai dan selesai dari pemrosesan *part* (*starting and completion time*).
Tujuan penjadwalan secara umum menurut Baker (1974) adalah sebagai berikut:

1. Meningkatkan produktivitas mesin, yaitu dengan mengurangi waktu mesin menganggur.
2. Mengurangi persediaan barang setengah jadi dengan jalan mengurangi jumlah rata-rata pekerjaan yang menunggu dalam antrian suatu mesin karena mesin tersebut sibuk.
3. Mengurangi keterlambatan suatu pekerjaan.

2.2 Penjadwalan Sistem Produksi *Job Shop*

Penjadwalan mempunyai metode yang berbeda-beda untuk setiap tipe sistem produksi karena setiap sistem mempunyai karakteristik yang berbeda satu dengan yang lainnya. Demikian pula dengan sistem produksi *job shop*. Ciri khas persoalan *job shop* adalah aliran pekerjaan dalam *shop* tidak searah (*non unidirectional*). Waktu proses dan *routing* dari sejumlah *job* yang akan dijadwalkan ke dalam suatu tabel matriks yang disebut matriks waktu proses dan matriks *routing*, kemudian hasil penjadwalan digambarkan dalam *Gantt Chart* (peta Gantt).

Dalam permasalahan *job shop*, *n job* harus diproses pada *m* mesin dengan asumsi-asumsi yang ditentukan. Setiap *job* mempunyai beberapa operasi yang harus diproses di mesin yang berbeda. Oleh karena itu diperlukan pengurutan (*sequencing*) dari tiap operasi pada masing-masing *job*, agar diperoleh performansi yang baik. Kriteria performansi yang baik sangat bergantung pada tujuan dan kebijakan manajemen. Sehingga dalam penjadwalan *job shop* diperlukan *input* berupa jumlah *job*, jumlah operasi dalam tiap *job*, dan urutan proses beserta mesin yang memprosesnya (*routing*).

2.3 *Variable Neighborhood Search*

Menurut Hansen dan Mladenovic (1997, 2003) VNS merupakan suatu *metaheuristic* yang digunakan dalam menyelesaikan sebuah permasalahan kombinasi dan optimisasi. Ide dasar dari VNS ini ialah memanfaatkan perubahan struktur yang terjadi dalam *neighborhood*. Hal tersebut berfungsi untuk melakukan pencarian solusi ketika pencarian solusi terjebak dalam minimum lokal. *Variable neighborhood descent* (VND) merupakan salah satu pengembangan konsep dasar yang diambil dari *variable neighborhood search* (VNS). Perbedaan antara VNS dan VND terletak pada perubahan *neighborhood*, VND melakukan perubahan *neighborhood* secara deterministik sedangkan VNS melakukan perubahan *neighborhood* secara acak (*random*).

Pada tahap awal VND dilakukan pemilihan satu set struktur *neighborhood* dan urutan dari setiap implementasinya telah ditentukan. Tahap selanjutnya adalah menetapkan sebuah solusi inisial yang fisibel sebagai *current solution*. Jika pada saat *looping* pencarian yang dimulai pada *neighborhood* pertama ditemukan sebuah hasil yang lebih baik dari *current solution*, maka solusi tersebut menggantikan *current solution* sebelumnya. Pencarian dimulai kembali pada *neighborhood* awal dengan *current solution* yang baru. Proses akan terus berlangsung hingga seluruh *neighborhood* dan tidak ditemukannya solusi yang lebih baik lagi dari *current solution*.

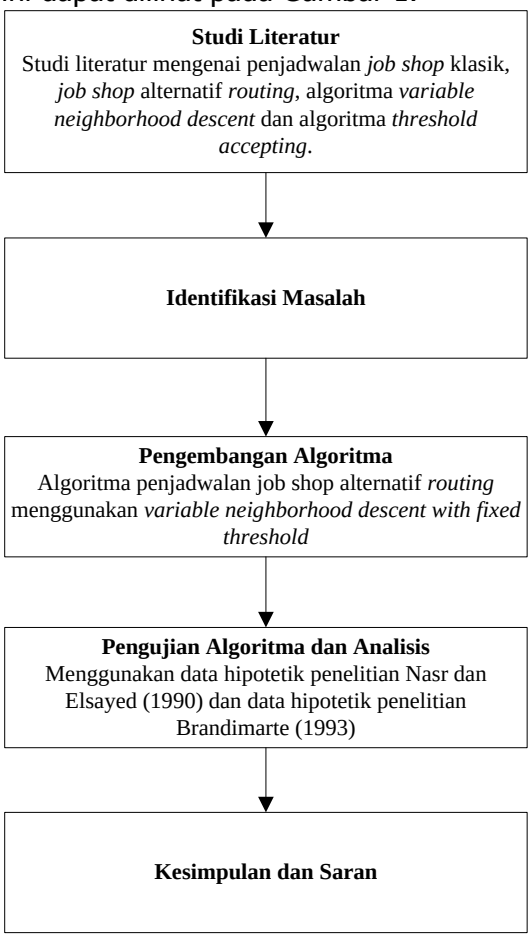
2.4 *Threshold Accepting*

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.

Pencarian dimulai dengan inisiasi jumlah iterasi yang dilakukan, jumlah step yang yang dibutuhkan dengan setiap iterasi untuk mengeksplorasi *neighborhood* dan nilai dari *threshold*. Nilai *threshold* selama pencarian proses dapat ditentukan dengan fungsi g_t, dimana t adalah jumlah iterasi. Selanjutnya mencari secara random solusi s' dari *neighborhood* $N(x)$ dari aliran solusi x. x' diterima jika $c(x')-c(x) \leq T_h$ dimana $T_h>0$. Nilai *Threshold* secara berangsur-angsur meningkat selama proses pencarian dan akan mendekati nol setelah akhir pencarian, ini menandakan hanya perpindahan yang memperbaiki yang diterima. Perubahan yang fleksibel dimana T_h juga diterima untuk meningkat atau untuk meng-reset juga ada.

3. METODOLOGI PENELITIAN

Metodologi penelitian merupakan langkah-langkah yang akan dilakukan dalam penelitian untuk mencapai tujuan yang diinginkan. Langkah-langkah pemecahan masalah dalam pengembangan algoritma ini dapat dilihat pada Gambar 1.



Gambar 1. Langkah-langkah Pemecahan Masalah

Peta posisi penelitian terhadap beberapa beberapa penelitian yang lalu dapat dilihat pada Gambar 2.

		Heuristik		
		VNS/VND	VNS/VND *with Fixed Threshold*	*Greedy Procedure*
		Minimisasi *Makespan*	Minimisasi *Makespan*	Minimisasi *Mean Flow Time*
Job Shop	Klasik	Sevkali & Aydin (2006)		
	Alternatif *Routing* / *Flexible Job Shop*	Benny (2009)	Penelitian	Nasr & Elsayed (1990)

Gambar 2. Peta Posisi Penelitian

4. PENGEMBANGAN ALGORITMA

Pengembangan Algoritma

Algoritma yang dikembangkan dalam penelitian ini merupakan pengembangan dari penelitian-penelitian sebelumnya, yaitu:

1. Nasr dan Elsayed (1990) yang membahas permasalahan *job shop* alternatif mesin dengan menggunakan *greedy procedure* untuk minimisasi *mean flow time*.
2. Sevkali dan Aydin (2006) mengusulkan penyelesaian permasalahan *job shop* menggunakan *variable neighborhood search* dengan kriteria minimisasi *makespan*.
3. Dueck dan Scheuer (1990) mengembangkan metode metaheuristik untuk pencarian solusi optimasi kombinatorial.

5. PENGUJIAN ALGORITMA DAN ANALISIS

5.1 Pengujian Algoritma

Skenario 1 adalah set data dari penelitian Nasr dan Elsayed (1990).

Tabel 1. Matriks Alternatif *Routing* Skenario 1

		Mesin					
		1	2	3	4	5	6
Job 1	O_{11}	2	3	4			
	O_{12}		3		2	4	
	O_{13}	1	4	5			
Job 2	O_{21}	3		5		2	
	O_{22}	4	3			6	
	O_{23}			4		7	11
Job 3	O_{31}	5	6				
	O_{32}		4		3	5	
	O_{33}			13		9	12
Job 4	O_{41}	9		7	9		
	O_{42}		6		4		5
	O_{43}	1		3			3

Skenario 2 adalah set data dari penelitian Brandimarte (1993).

Tabel 2. Matriks Alternatif *Routing* Skenario 2

		Mesin					
		1	2	3	4	5	6
Job 1	O_{11}	5		4			
	O_{12}		1	5		3	
	O_{13}			4			2
	O_{14}	1	6				5
	O_{15}			1			
	O_{16}			6	3		6
Job 2	O_{21}		6				
	O_{22}			1			
	O_{23}	2					
	O_{24}		6		6		
	O_{25}	1	6				5
Job 3	O_{31}		6				
	O_{32}			4			2
	O_{33}	1	6				5
	O_{34}		6	4			6
	O_{35}	1				5	
Job 4	O_{41}	1	6				5
	O_{42}		6				
	O_{43}			3			
	O_{44}		1	5		3	
	O_{45}			4			2
Job 5	O_{51}		1	5		3	
	O_{52}	1	6				5
	O_{53}		6				
	O_{54}	5		4			
	O_{55}		6		6		
	O_{56}		6	4			6

		Mesin					
		1	2	3	4	5	6
Job 6	O_{61}			4			2
	O_{62}	1					
	O_{63}		6	4			6
	O_{64}		6				
	O_{65}	1	6				5
	O_{66}	3			2		
Job 7	O_{71}						1
	O_{72}	3			2		
	O_{73}		6	4			6
	O_{74}	6	6			1	
	O_{75}			1			
Job 8	O_{81}			4			2
	O_{82}		6	4			6
	O_{83}	1	6				5
	O_{84}		6				
	O_{85}		6			6	
Job 9	O_{91}						1
	O_{92}	1				5	
	O_{93}			6	3		6
	O_{94}	2					
	O_{95}		6	4			6
	O_{96}		6		6		
Job 10	O_{101}			4			2
	O_{102}		6	4			6
	O_{103}		1	5		3	
	O_{104}						1
	O_{105}		6		6		
	O_{106}	3			2		

Notasi-notasi yang digunakan adalah sebagai berikut:

n = *jumlah job.*
m = *jumlah mesin.*
i = *indeks job (1,2,…,n)*
j = *indeks operasi job i.*
k = *mesin k (1,2,…,m)*
r_{ij} = *ready time operasi j job i siap untuk dijadwalkan.*
R_k = *ready time mesin k.*
t = *stage jadwal ke t.*
PS_0 = *jadwal parsial ke nol.*
S_t = *set operasi yang dapat dijadwalkan pada stage t, setelah diperoleh* PS_t.
σ_j = *waktu tercepat operasi* $j \in S_t$ *dapat dimulai.*
ϕ_j = *waktu tercepat operasi* $j \in S_t$ *dapat diselesaikan.*
C_{ijk} = *saat selesai job i operasi j di mesin k.*
t_{ijk} = *waktu proses job i operasi j di mesin k.*
MS = *makespan.*
MS^h = *makespan iterasi ke h.*
MS_{Th} = *makespan threshold.*
S = *jadwal terbaik.*
L_c = *operasi job yang membentuk lintasan kritis.*
SO_L = *himpunan operasi dari job yang membentuk lintasan kritis.*
O_b = *operasi terakhir pada jadwal yang telah dibuat.*
S^h = *jadwal setelah dilakukan proses insert/exchange untuk iterasi ke h.*
C_{ijk}^h = *saat selesai job i operasi j di mesin k, iterasi ke h.*

Tahap-tahap yang dilakukan pada algoritma usulan adalah sebagai berikut:

1. Tahap konstruksi

   Tahap konstruksi adalah tahap pembentukan inisial solusi dengan menggunakan metode penjadwalan *non delay.*

2. Tahap *Local search*

   Tahap ini adalah pencarian solusi yang lebih baik dengan cara merubah struktur *neighborhood.* Terdapat 2 struktur *neighborhood* pada algoritma usulan ini, yaitu:

   a. *Insert*

      Proses *insert* merupakan perubahan struktur dengan cara menyisipkan sebuah operasi ke dalam operasi lainnya.

   b. *Exchange*

      Proses *exchange* merupakan perubahan struktur dengan cara melakukan pertukaran posisi dari dua operasi yang terpilih pada satu stasiun kerja.

Tahap Konstruksi: penjadwalan dibuat menggunakan aturan penjadwalan *non delay* yang telah dimodifikasi.

Langkah 1

Input data: - matriks *routing* dan matriks waktu proses setiap *job*.

Set: - *Ready time* operasi pertama dari setiap *job* sama dengan nol.

$[r_{i1} = 0 \ \forall \ i = (1,2,...,n)]$

- *Ready time* seluruh mesin sama dengan nol.

$[R_k = 0 \ \forall \ k = (1,2,...,n)]$

- Tentukan $t = 0$

Langkah 2

Mulai dengan PS_0 sebagai jadwal parsial nol. Tentukan seluruh operasi tanpa *predecessor* untuk semua alternatif *routing* sebagai S_0.

Langkah 3

Tentukan $\sigma^* = \min_{j \in St}\{\sigma_j\}$ dan mesin m^* yaitu mesin tempat σ^* dapat direalisasikan.

Langkah 4

Untuk setiap operasi $j \in S_t$ yang membutuhkan mesin m^* dan berlaku $\sigma_j = \sigma^*$, buat jadwal parsial baru dengan menambahkan operasi j pada PS_t dengan saat mulai operasi pada σ_j.

Langkah 5

Untuk setiap jadwal parsial baru PS_{t+1}, yang dihasilkan pada Langkah 4, perbaharui (*update*) set data berikut:

- keluarkan operasi j dari S_t termasuk operasi pada *routing* alternatif.
- Tambahkan suksesor langsung operasi j ke dalam S_{t+1}.
- Naikkan nilai t dengan 1.

Langkah 6

Untuk setiap PS_{t+1} yang dihasilkan pada Langkah 4, kembali ke Langkah 3. Lanjutkan langkah-langkah ini sampai suatu jadwal *non delay* dihasilkan.

Langkah 7

Tampilkan *gantt chart* hasil penjadwalan.

Tahap *Local Search*: terdiri dari proses *insert* dan *exchange*.

Langkah 8

Nyatakan saat selesai tiap operasi *job* dan *makespan* yang dihasilkan dari penjadwalan di Langkah 7.

C_{ijk} = maks [r_{ij} , R_k] + t_{ijk} ; $\forall$ i = (1,2,…,n) (1)
MS = maks [C_{ijk}] (2)
Nyatakan jadwal yang terbentuk sebagai S dan tentukan L_c.

Langkah 9
Hitung nilai MS_{Th}.
MS_{Th} = MS terbaik + (MS terbaik x % Th) (3)

Langkah 10 (Proses *Insert*)
h = 1

Langkah 11
*Input*kan: S dan L_c
Nyatakan operasi terakhir yang berada pada lintasan kritis sebagai O_b. Masukkan operasi-operasi *job* terakhir yang berada pada lintasan ke dalam himpunan SO_L.

Langkah 12
Periksa O_b apakah dapat dilakukan *insert*.
Jika ya, lakukan *insert* sebanyak *job* yang berada pada posisi O_b sampai O_{b-1} pada operasi-operasi yang terdapat pada lintasan kritis (SO_L) dan lanjutkan ke Langkah 13.
Jika tidak, lanjutkan ke Langkah 16.

Langkah 13
Hitung saat selesai tiap operasi *job* dan *makespan*.
C^h_{ijk} = maks [r_{ij} , R_k] + t_{ijk} ; $\forall$ i = (1,2,…,n) (4)
MS^h = maks (C^h_{ijk}) (5)
Nyatakan jadwal yang terbentuk sebagai S^h dan tentukan L_c.

Langkah 14
Periksa $MS^h \leq MS_{TH}$
Jika ya, gunakan jadwal tersebut kemudian lanjutkan ke Langkah 15.
Tentukan L_c dan SO_L.
Set: h = h + 1.
Jika tidak, kembali ke Langkah 11.

Langkah 15
Periksa apakah $SO_L \neq \{\}$
Jika ya, kembali ke Langkah 11.
Lainnya, lanjutkan ke Langkah 16.

Langkah 16 (Proses *Exchange*)
*Input*kan: S dan L_c
Nyatakan operasi terakhir yang berada pada lintasan kritis sebagai O_b. Masukkan operasi-operasi *job* terakhir yang berada pada lintasan ke dalam himpunan SO_L.

Langkah 17
Periksa O_b apakah dapat dilakukan proses *exchange*.
Jika ya, lakukan proses *exchange* sebanyak *job* yang berada pada posisi O_b sampai O_{b-1} pada operasi-operasi yang terdapat pada lintasan kritis (SO_L) dan lanjutkan ke Langkah 18.
Jika tidak, lanjutkan ke Langkah 21.

Langkah 18
Lakukan proses seperti pada Langkah 13.

Langkah 19
Periksa $MS^h \leq MS_{TH}$
Jika ya, gunakan jadwal tersebut kemudian lanjutkan ke Langkah 20.
Tentukan L_c dan SO_L.
Set: $h = h + 1$.
Jika tidak, kembali ke Langkah 16.

Langkah 20
Periksa apakah $SO_L \neq \{\}$
Jika ya, kembali ke Langkah 16.
Lainnya, lanjutkan ke Langkah 21.

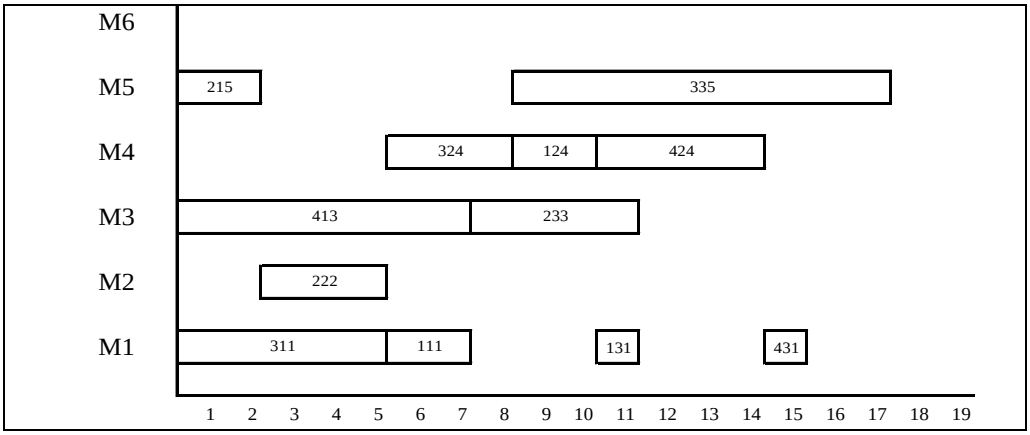
Langkah 21
Record jadwal yang memberikan *makespan* minimum dan tampilkan *Gantt Chart* hasil penjadwalannya.

Hasil yang diperoleh dari penelitian untuk Skenario 1 dapat dilihat pada Tabel 3.

Tabel 3. Hasil Penjadwalan Skenario 1

No	*Job*	Operasi	Mesin	Waktu Operasi	*Start Time*	*Finish Time*
1	1	1	1	2	5	7
2		2	4	2	8	10
3		3	1	1	10	11
4	2	1	5	2	0	2
5		2	2	3	2	5
6		3	3	4	7	11
7	3	1	1	5	0	5
8		2	4	3	5	8
9		3	5	9	8	17
10	4	1	3	7	0	7
11		2	4	4	10	14
12		3	1	1	14	15

Gantt chart hasil penjadwalan Skenario 1 dapat dilihat pada Gambar 3.



Gambar 3. *Gantt Chart* Hasil Penjadwalan Skenario 1

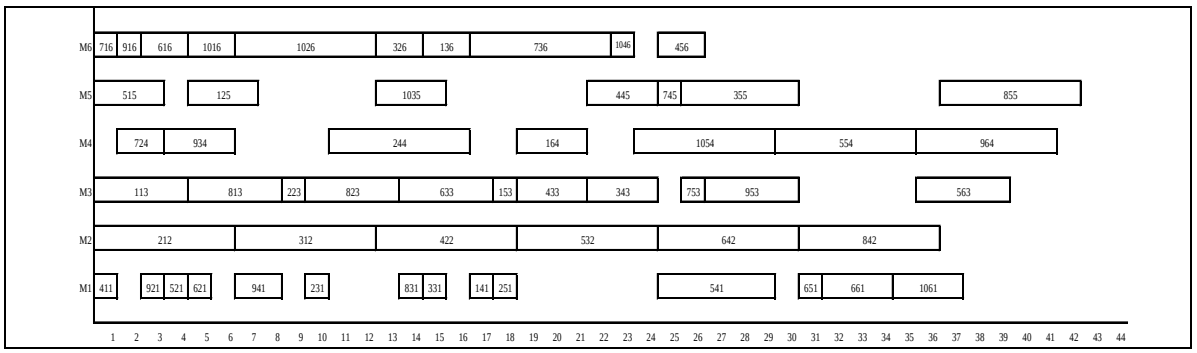
Hasil yang diperoleh dari penelitian untuk Skenario 2 dapat dilihat pada Tabel 4.

Tabel 4. Hasil Penjadwalan Skenario 2

Job	Operasi	Mesin	Waktu
Job 1	O_{11}	3	4
	O_{12}	5	3
	O_{13}	6	2
	O_{14}	1	1
	O_{15}	3	1
	O_{16}	4	3
Job 2	O_{21}	2	6
	O_{22}	3	1
	O_{23}	1	1
	O_{24}	4	6
	O_{25}	1	1
Job 3	O_{31}	2	6
	O_{32}	6	2
	O_{33}	1	1
	O_{34}	3	3
	O_{35}	5	5
Job 4	O_{41}	1	1
	O_{42}	2	6
	O_{43}	3	3
	O_{44}	5	3
	O_{45}	6	2
Job 5	O_{51}	5	3
	O_{52}	1	1
	O_{53}	2	6
	O_{54}	1	5
	O_{55}	4	6
	O_{56}	3	4

Job	Operasi	Mesin	Waktu
Job 6	O_{61}	6	2
	O_{62}	1	1
	O_{63}	3	4
	O_{64}	2	6
	O_{65}	1	1
	O_{66}	1	3
Job 7	O_{71}	6	1
	O_{72}	4	2
	O_{73}	6	6
	O_{74}	5	1
	O_{75}	3	1
Job 8	O_{81}	3	4
	O_{82}	3	4
	O_{83}	1	1
	O_{84}	2	6
	O_{85}	5	6
Job 9	O_{91}	6	1
	O_{92}	1	1
	O_{93}	4	3
	O_{94}	1	2
	O_{95}	3	4
	O_{96}	4	6
Job 10	O_{101}	6	2
	O_{102}	6	6
	O_{103}	5	3
	O_{104}	6	1
	O_{105}	4	6
	O_{106}	1	3

Gantt chart hasil penjadwalan Skenario 2 dapat dilihat pada Gambar 4.



Gambar 4. *Gantt Chart* Hasil Penjadwalan Skenario 2

Hasil perbandingan algoritma usulan dengan penelitian yang telah dilakukan sebelumnya dapat dilihat pada Tabel 5.

Tabel 5. Hasil Perbandingan Algoritma Usulan

Set Data	*n*	*m*	Nasr & Elsayed (1990)	Moon & Lee (2000)	Prassetiyo et al. (2004)	Pezzella et al. (2008)	Gao et al. (2008)	Benny (2009)	Algoritma Usulan
NE	4	6	18	17	17			17	17
MK-01	10	6				40	40	40	40

Keterangan: - NE = Set data pada penelitian Nasr dan Elsayed (1990)
- MK-01 = Set data pada penelitian Brandimarte (1993)

5.2 Analisis

Berdasarkan Tabel 5, dapat diketahui bahwa algoritma VND *with fixed threshold* memiliki keandalan dalam menyelesaikan permasalahan penjadwalan *job shop* alternatif *routing*. Algoritma VND *with fixed threshold* menghasilkan nilai *makespan* yang sama pada set data NE untuk penelitian Moon dan Lee (2000), Prassetiyo *et al*. (2004) dan Benny (2009). Pada ketiga penelitian tersebut, solusi inisial dibangkitakan berkali-kali secara *random*, sedangkan

pada algoritma usulan solusi inisial dibuat secara deterministik menggunakan aturan penjadwalan *non delay* sehingga waktu penyelesaian yang dihasilkan dapat lebih cepat. Pada penelitian Nasr dan Elsayed (1990), solusi inisial hanya dibangkitkan satu kali. Algoritma VND *with fixed threshold* menghasilkan *makespan* lebih baik 1 satuan waktu dibandingkan dengan penelitian Nasr dan Elsayed (1990).

Untuk set data MK-01, algoritma VND *with fixed threshold* menghasilkan nilai *makespan* yang sama dengan penelitian Pezzella *at al.* (2008), penelitian Gao *et al*. (2008) dan penelitian Benny (2009). Pada penelitian Pezzella *et al.* (2008) dan penelitian Benny (2009), inisial solusi dibangkitkan secara *random* sedangkan pada penelitian Gao *et al*. (2008) solusi inisial didapat menggunakan algoritma genetik. Perhitungan untuk set data MK-01 secara lengkap dapat dilihat pada Lampiran B. Berdasarkan hasil Skenario 2, maka algoritma usulan memiliki keandalan untuk menyelesaikan permasalahan *job shop* alternatif *routing*.

Selain menggunakan nilai *threshold* sebesar 10%, digunakan juga nilai *threshold* sebesar 5% dan 15%. Semuanya memberikan nilai *makespan* yang sama yaitu 17 satuan waktu untuk set data NE dan 40 satuan waktu untuk set data MK-01. Yang membedakan adalah pada tahap *local search*. Pada algoritma yang menggunakan nilai *threshold* sebesar 5% memberikan alternatif solusi lebih sedikit, sedangkan pada algoritma yang menggunakan nilai *threshold* sebesar 15% memberikan alternatif solusi lebih banyak. Hal ini terkait dengan nilai ambang batas penerimaan solusi.

6. KESIMPULAN

Kesimpulan yang diperoleh dari hasil penelitian adalah:

1. Pengembangan algoritma yang diusulkan adalah algoritma *variable neighborhood descent with fixed threshold* yang diuji untuk menyelesaikan permasalahan *job shop* alternatif *routing* dengan kriteria minimisasi *makespan*.
2. Algoritma usulan memungkinkan ruang penerimaan solusi yang lebih besar dibandingkan penelitian sebelumnya karena mempertimbangkan semua solusi yang muncul meskipun bukan solusi terbaik selama solusi tersebut berada dalam ambang batas penerimaan.
3. Nilai *fixed threshold* berpengaruh pada proses pencarian global optimum. Semakin bersar nilai *fixed threshold* semakin besar ruang penerimaan solusi, semakin kecil nilai *fixed threshold* semakin sempit ruang penerimaan solusi.

REFERENSI

Benny, 2009, *Model Penjadwalan Job Shop Alternatif Routing Menggunakan Variable Neighborhood Descent untuk Minimisasi Makespan*, Institut Teknologi Nasional, Bandung.

Brandimarte, P., 1993, Routing and Scheduling in a Flexible Job Shop by Tabu Search. *Annuals of Operation Research* 41, 157-183.

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.

Mladenovic, N. dan Hansen, P., 1997, *Variable Neighborhood Search*, Computer & Operation Research 24, 1097-1100.

Nasr, N. dan Elsayed, E. A., 1990, Job Shop Scheduling with Alternative Machines, *International Journal of Production Research* 28, No. 29, 1595-1609.

Sevkali, M. dan Aydin, M. E., 2006, Variable Neighborhood Search for Job Shop Scheduling Problems, *Journal of Software* 1, No. 2, 34-39.