Algoritma *Variable Neighbourhood Descent with Fixed Threshold* untuk Keseimbangan Lintasan Perakitan Tunggal dengan Kriteria Minimisasi Jumlah Stasiun Kerja*

SEPTA HARDINI, EMSOSFI ZAINI, ARIF IMRAN

Jurusan Teknik Industri
Institut Teknologi Nasional (ITENAS) Bandung

Email: septahardini@yahoo.com

ABSTRAK

Penelitian ini membahas simple assembly line balancing problem tipe I (SALBP I) menggunakan algoritma variable neighbourhood descent with fixed threshold dengan kriteria minimisasi jumlah stasiun kerja. Algoritma variable neighbourhood descent with fixed threshold ini terdiri atas dua tahap, yaitu tahap pembangkitan solusi inisial dan tahap local search. Prinsip dasar dari algoritma ini ialah dengan melakukan pencarian solusi pada neighbourhood yang terbentuk dari sekumpulan elemen kerja dengan melakukan modifikasi atau perubahan struktur neighbourhood dengan mempertimbangkan $\%T_h$ serta solusi yang bukan solusi terbaik dalam algoritmanya. Performansi algoritma usulan diuji dengan menggunakan 3 set data dari data literatur. Hasil pengujian algoritma usulan tersebut memberikan solusi yang lebih baik dibandingkan dengan hasil penelitian-penelitian sebelumnya yang telah dipublikasikan.

Kata kunci: *simple assembly line balancing problem, lintasan perakitan tunggal, minimisasi stasiun kerja*

ABSTRACT

This study discusses the simple assembly line balancing problem of type I (SALBP I) using variable neighbourhood descent algorithm with fixed threshold by considering minimum number of work stations criteria. Variable neighbourhood descent algorithm with fixed threshold consists of two stages, namely initial solution generation stage and local search stage. The basic principle of this algorithm is to perform the search for solutions to the neighbourhood made up of a set of elements of the work by making modifications or changes to the structure by considering the neighbourhood %Th, and the solution is not the best solution in the algorithm. Performance of the proposed algorithm was tested using three sets of data from the literature data. Test results of proposed algorithm gives better solutions with the results of previous studies that have been published.

* *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*

Keywords: *simple assembly line balancing problem, single-track assembly, minimizationwork station*

1. PENDAHULUAN

Keseimbangan lintasan perakitan (*assembly line balancing,* ALB) merupakan suatu proses penempatan sejumlah elemen kerja (*task*) ke dalam beberapa stasiun kerja (*work station*) tanpa melanggar *precedence relations* dan waktu perakitan di setiap stasiun kerja tidak melebihi waktu siklus lintasan perakitan tersebut. Tujuan yang ingin dicapai adalah menghasilkan konfigurasi elemen kerja dalam stasiun kerja yang memberikan Efisiensi Lintasan (EL) yang tinggi dan *Smoothness Index* (SI) yang rendah.Lintasan perakitanadalah serangkaian stasiun kerja yang secara berurutan digunakan untuk merakit produk dimulai dari komponen-komponen sampai menjadi suatu produk jadi, dimana *part-part* (komponen-komponen) dirakit pada sebuah *part* dasar/*part* utama diatas sebuah kendaraan pembawa/ban berjalan (*conveyor*). Lintasan produksi yang efisien adalah lintasan produksi yang dapat memenuhi target laju *output* dengan jumlah stasiun kerja yang minimum atau jumlah operator minimum. Lintasan perakitan yang digunakan perusahaan biasanya menggunakan satu lintasan perakitan atau lebih. Menurut Scholl (1999) lintasan perakitan model tunggal merupakan *layout* tradisional dari lintasan perakitan.

Algoritma *Variable Neighbourhood Descent* (VND) merupakan metode metaheuristik yang diterapkan oleh Hansen dan Mladenovic (1999). Prosedur pada VND dimulai dengan melakukan perbaikan pada *neighbourhood* pertama, jika minimum lokal sudah tercapai maka perbaikan dilakukan pada *neighbourhood* kedua dan seterusnya. Rahadian (2010) menggunakan algoritma VND pada penelitiannya untuk menyelesaikan SALBP I dengan kriteria minimisasi jumlah stasiun kerja. Selain VND, metode metaheuristik lainnya adalah *t Threshold Accepting* (TA) yang diperkenalkan oleh Dueck dan Scheuer (1990) untuk menyelesaikan masalah optimisasi kombinatorial. Dalam metode ini, terdapat ambang batas toleransi untuk menerima solusi inisial yang masih dalam ambang batas dan lebih baik dari solusi awal untuk dieksplorasi sehingga menemukan solusi yang lebih baik. Fungsi batas *threshold* tersebut dapat diadaptasikan ke algoritma yang lain. Berdasarkan penelitian-penelitian sebelumnya, penggabungan algoritma VND dengan fungsi *threshold* belum dikembangkan sehingga perlu dilakukan penelitian pengembangan algoritma VND *with fixed threshold*. Oleh karena itu pada penelitian ini dikembangkan model SALBP I menggunakan algoritma VND *with fixed threshold* dengan kriteria minimisasi jumlah stasiun kerja.

Penelitian ini membahas mengenai *simple assembly line balancing problem* Tipe I (SALBP I). Setiap elemen kerja akan ditempatkan pada sejumlah stasiun kerja tanpa melanggar *precedence relations* dan beban kerja di setiap stasiun kerja tidak melebihi waktu siklus sehingga efisiensilintasan yang dihasilkan dapat maksimal. Algoritma yang digunakan sebagai alat pemecahan masalah adalah algoritma VND *with fixed threshold* dan ukuran performansi yang digunakan adalah minimisasi jumlah stasiun kerja dengan memaksimumkan efisiensi lintasan.

Penelitian ini bertujuan untuk menghasilkan algoritma VND *with fixed threshold* dengan kriteria minimisasi jumlah stasiun kerja.Beberapa pembatas yang digunakan adalah:

1. Jumlah operator di setiap stasiun kerja tidak diperhitungkan.
2. Peralatan atau mesin yang digunakan untuk merakit produk tidak dibahas.

Asumsi-asumsi yang digunakan dalam penelitian ini adalah:
1. Waktu operasi setiap elemen kerja telah diketahui secara deterministik.
2. Waktu siklus lintasan perakitan telah diketahui secara deterministik.
3. Waktu *set up* setiap elemen kerja diasumsikan termasuk ke dalam waktu operasi setiap elemen kerja.

2. METODOLOGI PENELITIAN

2.1.Studi Literatur

Pada tahap ini ditinjau beberapa literatur yang berhubungan dengan *Simple Assembly Line Balancing Problem* (SALBP), *Variable Neighbourhood Descent* (VND) dan *Threshold Accepting* (TA). Studi literaturnya antara lain:
1. Scholl (1999) mendefinisikan permasalahan dalam *Assembly Line Balancing* (ALB) menjadi *Simple Assembly Line Balancing Problem* I (SALBP I) dan *Simple Assembly Line Balancing Problem*II (SALBP II).
2. Hansen dan Mladenovic (1999) mengembangkan metoda *Variable Neighbourhood Descent*(VND) yang merupakan metode metaheuristik untuk memecahkan masalah-masalah NP *hard*.
3. Rahadian (2010) menyelesaikan SALBP I untuk lintasan perakitan tunggal dengan algoritma VND dengan kriteria minimisasi jumlah stasiun kerja.
4. Dueck dan Scheuer (1990) melakukan penelitian tentang *Threshold Accepting*(TA) untuk menyelesaikan masalah optimasi kombinatorial.

2.2.Identifikasi Posisi Penelitian

Simple Assembly Line Balancing Problem(SALBP) merupakan salah satu permasalahan klasik dalam permasalahan keseimbangan lintasan perakitan. SALBP dibagi menjadi 2 jenis yaitu SALBP I dan SALBP II. Metode optimasi, heuristik, dan metaheuristik telah banyak digunakan untuk menyelesaikan SALBP. Algoritma VND dengan nilai *threshold* tetap merupakan suatu metode metaheuristik untuk menyelesaikan masalah-masalah optimasi kombinatorial. Berdasarkan studi literatur,algoritma VND *with fixed threshold*diterapkan dalam permasalahan ALB khususnya SALBP I.

Berdasarkan hal tersebut, maka perlu dilakukan penerapan algoritma VND *with fixed threshold* untuk menyelesaikan SALBP I dengan kriteria minimisasi jumlah stasiun kerja. Peta posisi penelitian ini terhadap penelitian-penelitian lain dapat dilihat pada Gambar 1.

Keseimbangan Lintasan Perakitan		Minimisasi Jumlah Stasiun Kerja					
				Metaheuristik			
		Optimisasi	Heuristik	*Genetic Algorithm*	GRASP	VND	VND *with fixed threshold*
	SLABP I	Johnson (1988) Hoffman (1992) Scholl and Klein (1997)	Baybars (1986) Talbot *et al.* (1986) Boctor (1995) Ponnambalan *et al.* (1989)	Suresh *et al.* (1996) Sabuncuoglo *et al.* (2000) Goncalves dan De Almeida (2002)	Andres *et al.* (2006)	Rahadian (2010)	Penelitian

Gambar 1. Peta Posisi Penelitian

2.3.Pengembangan Algoritma

Penelitian ini bertujuan menghasilkan model keseimbangan lintasan perakitan sederhana I (SALBP I) menggunakan algoritma *Variable Neighbourhood Descent with Fixed Threshold*tetap dengan kriteria minimisasi jumlah stasiun kerja. Penelitian ini merupakan

pengembangan dari beberapa penelitian sebelumnya, diantaranya adalah:

1. Rahadian (2010) menyelesaikan SALBP I untuk lintasan perakitan tunggal dengan algoritma VND dengan kriteria minimisasi jumlah stasiun kerja. Algoritma ini merupakan pendekatan metaheuristik yang menggunakan set *neighbourhood* dalam proses pencarian solusi untuk menemukan *localoptimum* dengan melakukan perubahan *neighbourhood* seperti *exchange* dan *insert* secara deterministik sehingga dapat mencapai global optimum.
2. Dueck dan Scheuer (1990) mengembangkan metoda yang merupakan modifikasi dari simulasi *annealing* dimana menggunakan fungsi *probabilistic* dalam menerima *nonimproving* solusi. Dalam metode ini semua solusi yang terbaik diterima dan yang lebih rendah, yang memiliki nilai lebih kecil atau sama dengan batas ambang juga diterima.

Penelitian ini menggunakan algoritma VND *with fixed threshold* dalam menyelesaikan SALBP I untuk lintasan perakitan tunggal dengan kriteria minimisasi jumlah stasiun kerja.

2.4. Pengujian Algoritma dan Analisis

Pengujian algoritma ini dilakukan untuk mengetahui performansi dari model usulan dengan menggunakan tiga skenario. Untuk mengevaluasi performansi model usulan, maka hasil pengujian dibandingkan dengan beberapa peneltian yang telah dipublikasi.

2.5.Kesimpulan dan Saran

Bab ini berisi kesimpulan yang diperoleh dari hasil pengembangan model dan saran-saran yang ditujukan untuk pengembangan selanjutnya.

3. PENGEMBANGAN ALGORITMA

Notasi-notasi yang digunakan pada algoritma usulan adalah sebagai berikut:

i = indeks untuk stasiun kerja (SK); (i = 1, 2, …, m)
j =indeks untuk elemen kerja (EK); (j = 1, 2, …, n)
t_j = waktu proses pada elemen kerja ke-j
e = indeks iterasi pada proses *exchange* (e = 1, 2, 3, …, e_{maks})
r = indeks iterasi pada *insert* (r = 1, 2, 3, …, r_{maks})
k = indeks iterasi keseluruhan (k = 1,2,3, ..., $\lceil n/2 \rceil$)
CT = waktu siklus
ST_i = akumulasi waktu elemen kerja pada stasiun kerja ke-i
IT_i =*idle time* pada stasiun kerja ke-i
IT_{maks} =*idle time* dengan nilai terbesar
NY_i =*neighbourhood structure* penukar
NX_i =*neighbourhood structure* yang ditukar
NI_i =*neighbourhood structure* pada proses *insert*
VY_j =elemen kerja penukar
VX_j =elemen kerja ditukar
SI =*smoothness index*
EL = efisiensi lintasan
EL_{T_h} =batas bawah nilai efisiensi berdasarkan $\%T_h$
($ELT_h = EL_{currentsolution} - (EL_{currentsolution} \times \%T_h)$)
% Th = persentase batas *threshold*

Ide dasar dari pengembangan algoritma pada penelitian ini dapat dijelaskan sebagai berikut:

a. Tahap 1 *Initial Solution*

Pada tahap ini dilakukan pembentukan sebuah solusi inisial yang menggunakan algoritma

Region Approach (Elsayed & Boucher, 1994). Algoritma ini memiliki tahapan sebagai berikut:

- Langkah 1
  Kelompokkan seluruh elemen kerja ke dalam kolom, kolom I merupakan kolom yang berisi elemen kerja yang tidak menjadi elemen kerja pengikut (*succesor*) terhadap elemen kerja lainnya. Kolom II merupakan kolom yang berisi elemen kerja pengikut terhadap elemen kerja yang berada di kolom I, begitu pula untuk kolom-kolom selanjutnya sampai semua elemen kerja dikelompokkan ke dalam kolom masing-masing.
- Langkah 2
  Tentukan waktu siklus (untuk SALBP I waktu siklus telah ditetapkan/diketahui).
- Langkah 3:
  Tempatkan elemen kerja ke dalam stasiun kerja tanpa melebihi waktu siklus.
- Langkah 4:
  Jika pada saat proses penempatan terdapat elemen kerja yang melebihi waktu siklus, tempatkan elemen kerja tersebut pada stasiun kerja selanjutnya.
- Langkah 5:
  Ulangi langkah 3 sampai dengan 4 sehingga semua elemen kerja ditempatkan ke stasiun kerja masing-masing.

Setelah itu dilakukan penghitungan nilai *idle time* (*IT*), efisiensi lintasan (*EL*), dan *smoothness index* (*SI*) dengan menggunakan persamaan sebagai berikut:

Idle time (*IT*)

$$IT = CT - ST_i \quad (1)$$

Efisiensi Lintasan (EL)

$$EL = \frac{\sum_{i=1}^{m} ST_i}{(m)(CT)} \times 100\% \quad (2)$$

smoothness index (*SI*)

$$SI = \sqrt{\sum_{i=1}^{m} (ST_{maks} - ST_i)^2} \quad (3)$$

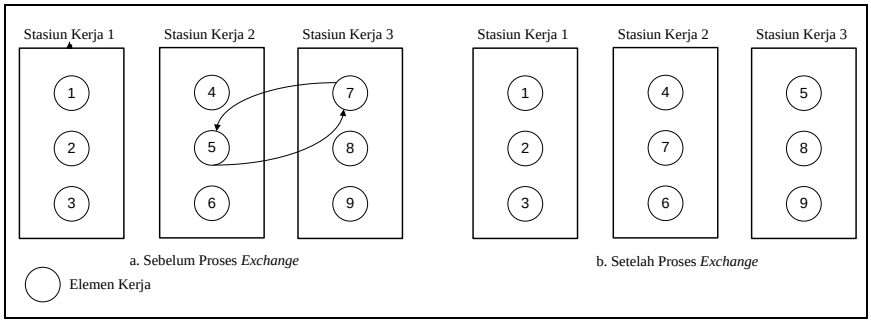
Konfigurasi nilai *IT*, *EL*, dan *SI* yang didapat pada tahap *initial solution* disimpan sebagai *current solution* (CS).

b. Tahap 2 Local Search

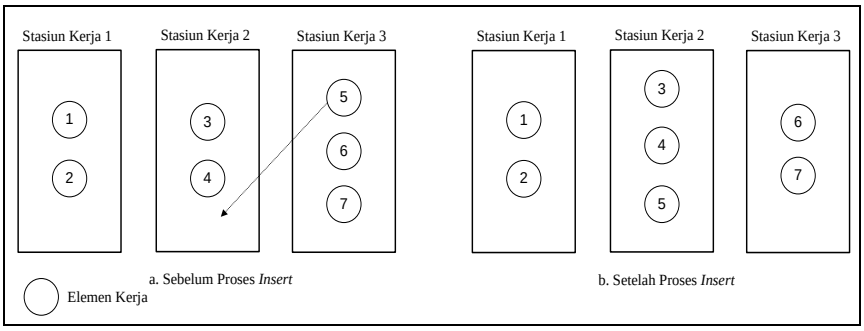
Pada tahap ini dilakukan pencarian solusi pada *neighbourhood* yang terbentuk dari sekumpulan elemen kerja dengan melakukan modifikasi atau perubahan struktur *neighbourhood*. Perubahan struktur *neighbourhood* ini dibagi menjadi 2 tahap yaitu:

- *Exchange*
  Proses *exchange* merupakan perubahan struktur dengan cara melakukan pertukaran posisi dari dua elemen kerja yang terpilih pada dua stasiun kerja yang berbeda. Sebagai contoh, misalnya terdapat 9 elemen kerja {1,2,3,4,5,6,7,8,9} yang dibagi kedalam 3 stasiun kerja. Stasiun kerja 1 dengan anggota elemen kerja {1,2,3}, stasiun kerja 2 dengan anggota elemen kerja {4,5,6} dan stasiun kerja 3 dengan elemen kerja {7,8,9}. Dari 9 elemen kerja tersebut terpilih elemen kerja 5 dan 7 yang dapat ditukar, pertukaran ini dapat dilakukan dengan tanpa melanggar *precedence constraint* dan tidak melebihi waktu siklus stasiun kerja.
  Contoh proses *exchange* yang terjadi pada 3 stasiun kerja dengan 9 elemen kerja dapat dilihat pada Gambar 2.

- *Insert*
  Proses *insert* merupakan perubahan struktur dengan cara menyisipkan sebuah operasi kedalam operasi lainnya. Sebagai contoh, misalnya terdapat 7 elemen kerja {1,2,3,4,5,6,7} yang terbagi ke dalam 3 stasiun kerja. Stasiun Kerja 1 dengan anggota elemen kerja {1,2}, stasiun kerja 2 dengan anggota elemen kerja {3,4} dan Stasiun Kerja 3 dengan anggota elemen kerja {5,6,7}. Berdasarkan ke 7 elemen kerja tersebut terpilih elemen kerja 5 yang dapat dilakukan proses *insert* dari Stasiun Kerja 3 ke Stasiun Kerja 2.
  Contoh proses *insert* yang terjadi pada 3 stasiun kerja dengan 7 elemen kerja dapat dilihat pada Gambar 3.



Gambar 2. Proses *Exchange* Pada Perubahan Suatu Struktur *Neighbourhood*



Gambar 3.Proses *Insert* pada Perubahan suatu Struktur *Neighbourhood*

Langkah-langkah algoritma VND *with fixed threshold* untuk menyelesaikan SALBP I dengan kriteria minimisasi jumlah stasiun kerja adalah sebagai berikut:

Tahap 1.*Initial Solution*

Langkah 0

Input data *j*, t_j, *CT*, *precedence relation*, dan *precedence diagram.*

Langkah 1

Bentuk solusi inisial dengan menggunakan algoritma *Region Approach* dan simpan sebagai *current solution* (CS).

Langkah 2

Setelah itu dilakukan penghitungan nilai *idle time* (*IT*), efisiensi lintasan (*EL*), dan *smoothness index* (*SI*).

Tahap 2. *Local Search* menggunakan *Variable Neighbourhood Descent* (VND) *with Fixed Threshold*

Langkah 3

Inputkan data *j*, t_j, *CT*, *EL*, *SI, ST*dari solusi inisial yang telah terbentuk dan % *Th*.

Set $k = 1$ dan $k_{maks} = \lceil {}^{n}/_{2} \rceil$.

Langkah 4
Set $e = 1$ dan $e_{maks} = m$.
Langkah 5
Tentukan IT setiap stasiun kerja dengan persamaan $IT_i = (CT - ST_i)$.
Langkah 6
Pilih nilai IT yang terbesar (IT_{maks}) dengan menggunakan persamaan $IT_{maks} = \overset{maks}{\forall_i}\{IT_i\}$.
Set NX_i dengan i yang terpilih berdasarkan nilai IT terbesar.
Jika terdapat nilai IT_i yang sama maka pilih stasiun kerja secara random, lanjutkan ke langkah 7.
Langkah 7
Apakah jumlah $j \in NX_i = 1$?
Jika ya, set j terpilih sebagai VX_j dan lanjutkan ke langkah 9.
Jika tidak, lanjutkan ke langkah 8.
Langkah 8
Pilih VX_j, $j \in NX_i$ menggunakan persamaan $VX_j = arg \underset{j \in NXi}{maks}\{t_j\}$ dan set j dengan nilai t terbesar sebagai VX_j, Jika terdapat t_j yang sama maka pilih secara random dan lanjutkan ke langkah 9.
Langkah 9
Pilih VX_j, $j \notin NX_i$ dan lanjutkan ke langkah 9.1.
9.1. Apakah proses *exchange* melanggar *precedence constraint* atau VY_j sudah pernah menghasilkan konfigurasi yang sama?
Jika ya, lanjutkan ke langkah 9.2.
Jika tidak, lanjutkan ke langkah 9.3.
9.2. Apakah masih terdapat $j \in NX_i$ yang belum ditukar?
Jika ya, kembali ke langkah 8.
Jika tidak, set $e = e + 1$ dan lanjutkan ke langkah 10.
9.3. Lakukan proses *exchange* untuk semua j yang memungkinkan untuk dilakukan proses *exchange*, hitung nilai *EL* dan *SI.*
Apakah $EL \geq EL_{T_h}$?
Jika ya, lanjutkan ke langkah 9.4.
Jika tidak, kembali ke langkah 9.2.
9.4. Apakah $|VY_j| > 1$?
Jika ya, pilih j dengan nilai *EL* terbesar, jika terdapat *EL* yang sama pilih *SI* terkecil, jika terdapat nilai *SI* yang sama pilih secara random dan lanjut ke langkah 9.5.
Jika tidak, lanjut ke langkah 9.5.
9.5. Apakah *EL* = 100%?
Jika ya, lanjutkan ke langkah 14.
Jika tidak, lanjutkan ke langkah 11.
Langkah 10
Apakah $e > e_{maks}$?
Jika ya, lanjutkan ke langkah 11.
Jika tidak, kembali dan ulangi langkah 5.
Langkah 11
Set $r = 1$ dan $r_{maks} = m$.
Set $i = r$ sebagai NI_i.
Langkah 12
Pilih $j \notin NI_i$ dan lanjutkan ke Langkah 12.1.
12.1. Apakah terdapat j yang dapat dilakukan proses *insert* tanpa melanggar *precedence constraint* terhadap NI_i?
Jika ya, lanjutkan ke langkah 12.2.

Jika tidak, set $r = r + 1$ dan lanjutkan ke langkah 12.3.

12.2. Apakah j yang di*insert* terhadap NI_i sudah pernah menghasilkan konfigurasi yang sama?
Jika ya, set $r = r + 1$ dan lanjutkan ke langkah 12.3.
Jika tidak, lanjutkan ke langkah 12.4.

12.3. Apakah $r = m + 1$?
Jika ya, lanjutkan ke langkah 13.
Jika tidak, kembali dan ulangi langkah 12.

12.4. Lakukan proses *insert* untuk semua *j* yang memungkinkan.
Apakah jumlah stasiun kerja berkurang?
Jika ya, lanjutkan ke langkah 12.6.
Jika tidak, hitung nilai *EL* dan *SI*.
Apakah $EL \geq EL_{T_h}$?
Jika ya, lanjutkan ke langkah 12.5.
Jika tidak, set $i = i + 1$ dan lanjutkan ke langkah 12.3.

12.5. Apakah terdapat lebih dari 1, j yang dapat dilakukan proses *insert*?
Jika ya, pilih *j* dengan nilai *EL* terbesar, jika terdapat *EL* yang sama pilih *SI* terkecil, jika terdapat nilai *SI* yang sama pilih secara random dan lanjut ke langkah 12.6.
Jika tidak, lanjutkan ke langkah 12.6.

12.6. Apakah *EL* = 100%?
Jika ya, lanjutkan ke langkah 14.
Jika tidak, lanjutkan ke langkah 13.

Langkah 13

Pilih konfigurasi dengan jumlah SK terkecil dari *current solution*, *exchange,* dan *insert,* jika jumlah SK sama pilih nilai *EL* terbesar jika *EL* sama pilih *SI* terkecil, jika nilai *SI* sama pilih *current solution.*

Kemudian set konfigurasi terpilihsebagai *current solution*.

Set *k = k +*1.

Apakah *k* >$\lceil n/2 \rceil$?

Jika ya, lanjutkan ke langkah 14

Jika tidak kembali dan ulangi langkah 4.

Langkah 14

Simpan dan tampilkan konfigurasi terbaik penugasan seluruh elemen kerja (*j)* pada tiap stasiun kerja (*i).*

4. PENGUJIAN ALGORITMA DAN ANALISIS

Algoritma usulan digunakan untuk menguji cara kerja algoritma usulan dengan menggunakan data dari penelitian Mitchell (1957) yang terdapat pada Scholl (1993), selain itu juga menggunakan data Rosenberg dan Ziegler (1992) dan Jackson (1956) yang juga terdapat pada Scholl (1993). Pengujian algoritma usulan dengan metode *variable neighbourhood descent with fixed threshold* menggunakan $\%T_h$ untuk menentukan ambang batas diterimanya nilai efisiensi, pada pengujian algoritma ini digunakan $\%T_h$diantaranya 5%, 10% dan 15%. Pengujian algoritma ini dilakukan dengan menggunakan tiga skenario. Skenario yang digunakan pada penelitian ini adalah:

a. Skenario 1

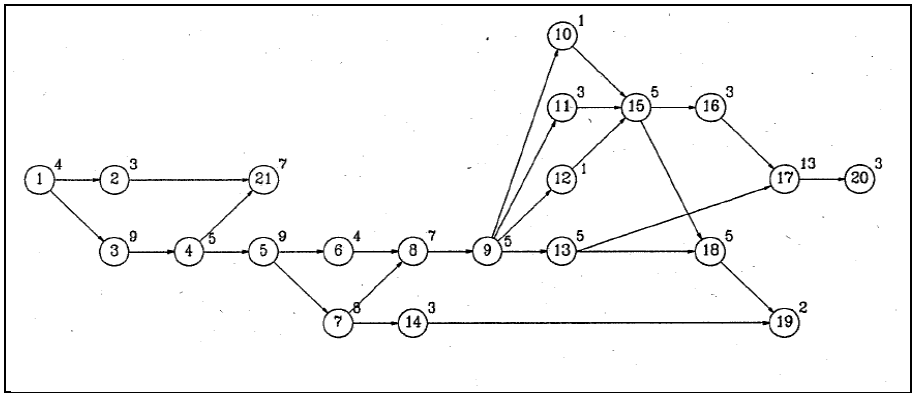
Skenario 1 menggunakan set data dari Mitchell (1957) dengan jumlah elemen kerja (*n*) = 21, dan waktu siklus (*CT*) = 26. Set data ini dapat dilihat pada Tabel 1 dan *precedence* diagramnya ditunjukkan pada Gambar 4.

b. Skenario 2

Skenario 2 menggunakan set data Rosenberg dan Ziegler (1992) memiliki jumlah elemen kerja (n) = 25, dan waktu siklus (CT) = 25. *Precedence relation* dan *precedence* diagram set data ini dapat dilihat pada Tabel 2 dan Gambar 5.

Tabel 1. Set Data dan *Precedence Relation* untuk Set Data Mitchell (1957)

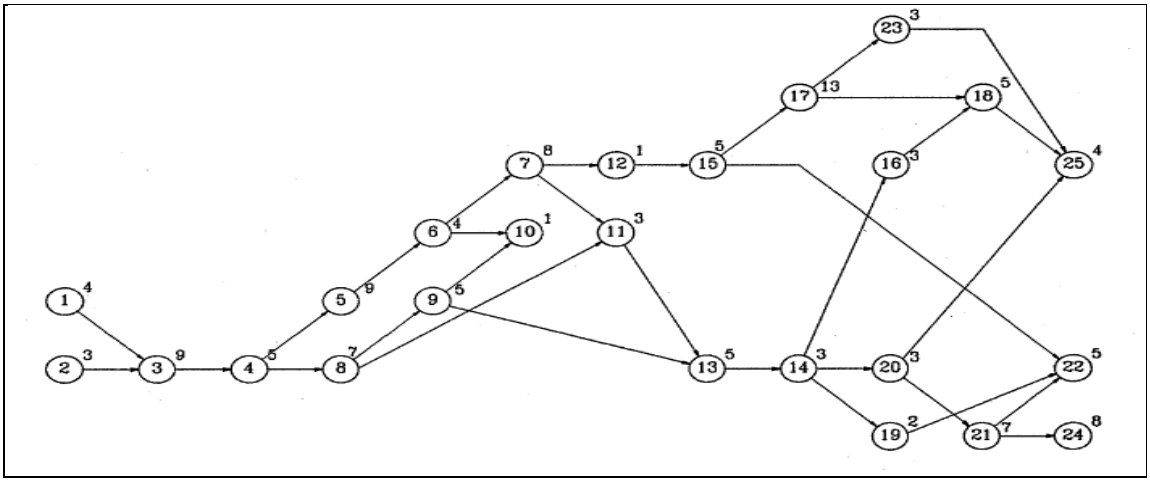
j	*Precedence Relation*	t_j
1	1,2;1,3	4
2	2,21	3
3	3,4	9
4	4,5	5
5	5,6;5,7	9
6	6,8	4
7	7,8;7,14	8
8	8,9	7
9	9,10;9,11;9,12;9,13	5
10	10,15	1
11	11,15	3
12	12,15	1
13	13,17;13,18	5
14	14,19	3
15	15,16;15,18	5
16	16,17	3
17	17,20	13
18	18,19	5
19	-	2
20	-	3
21	-	7



Gambar 4.*Precedence Diagram* Set Data Mitchel (1957)

Tabel 2. Set Data dan *Precedence Relation* untuk Set Data Rosenberg dan Ziegler (1992)

j	*Precedence Relation*	t_j
1	1,3	4
2	2,3	3
3	3,4	9
4	4,5;4,8	5
5	5,6	9
6	6,7;6,10	4
7	7,1;7,12	8
8	8,9;8,11	7
9	9,10;9,13	5
10	-	1
11	11,13	3
12	12,15	1
13	13,14	5
14	14,16;14,19;14,20	3
15	15,17;15,22	5
16	16,18	3
17	17,18	13
18	18,25	5
19	19,22	2
20	20,21;20,25	3
21	21,22;21,24	7
22	-	5
23	23,25	3
24	-	8
25	-	4



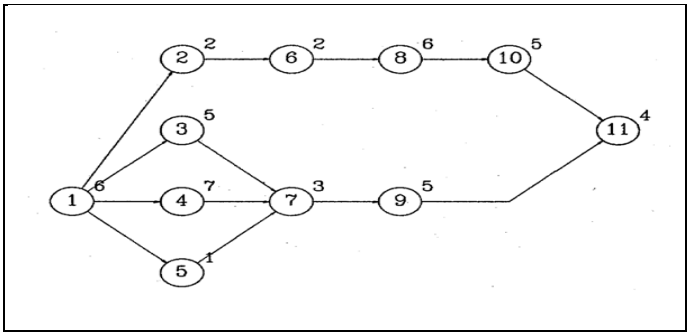
Gambar 5.*Precedence Diagram* Set Data Rosenberg dan Ziegler (1992)

c. Skenario 3

Skenario 3 menggunakan set data Jackson (1956) memiliki jumlah elemen kerja (n) = 11, dan waktu siklus (CT) = 14. Set data Jackson (1956) dapat dilihat pada Tabel 3 dan *precedence* diagramnya ditunjukkan Gambar 6.

Tabel 3. Set Data dan *Precedence Relation* untuk Set Data Jackson (1956)

j	*Precedence Relation*	t_j
1	1,2;1,3;1,4;1,5	6
2	2,6	2
3	3,7	5
4	4,7	7
5	5,7	1
6	6,8	2
7	7,9	3
8	8,10	6
9	9,11	5
10	10,11	5
11	-	4



Gambar 6.*Precedence Diagram* Set Data Data Jackson (1956)

Hasil perhitungan algoritma usulan untuk %T_h diantaranya 5%, 10% dan 15% dengan hasil penelitian sebelumnya dapat dilihat pada Tabel 4 dan Tabel 5.

Berdasarkan Tabel 4, %T_h yang diuji yaitu 5%, 10% dan 15% dengan menggunakan

algoritma usulan menghasilkan efisiensi lintasan yang sama. Jika dilihat dari ketiga $\%T_h$yang diuji, semakin besar $\%T_h$ maka akan semakin besar pula batasan efisiensi lintasan yang diterima dari hasil efisiensi lintasan *current solution.* Oleh sebab itu $\%T_h$ yang terpilih adalah 15% karena semakin besar batasan $\%T_h$ maka semakin banyak kemungkinan untuk dilakukannya tahapan *local search* sehingga kemungkinan untuk naiknya efisiensi lintasan ataupun berkurangnya jumlah stasiun kerja juga akan semakin besar.

Tabel 4.Hasil Perhitungan Algoritma Usulan untuk Persentase Batas *threshold* 5%, 10% dan 15%

Set Data	Effisiensi Lintasan				
	Jumlah Elemen Kerja (*n*)	Waktu Siklus (*CT*)	Pengujian Algoritma		
			%Th = 5%	%Th = 10%	%Th = 15%
Skenario 1	21	26	87,5%	87,5%	87,5%
Skenario 2	25	25	83,33%	83,33%	83,33%
Skenario 3	11	14	88,46%	88,46%	88,46%

Tabel 5. Perbandingan Hasil Perhitungan Algoritma Usulan dengan Hasil Penelitian Sebelumnya

Set Data	Jumlah Elemen Kerja (*n*)	Minimisasi Stasiun Kerja											
		Hoffman (1992)		Scholl (1997)		Goncalves dan Raimundo (2002)		Rahadian (2010)			Pengujian Algoritma (%Th = 15%)		
		Waktu Siklus (*CT*)	Jumlah Stasiun Kerja	Waktu Siklus (*CT*)	Jumlah Stasiun Kerja	Waktu Siklus (*CT*)	Jumlah Stasiun Kerja	Waktu Siklus (*CT*)	Jumlah Stasiun Kerja	EL	Waktu Siklus (*CT*)	Jumlah Stasiun Kerja	EL
Skenario 1	21	26	5	26	-	26	5	26	5	80,77%	24	5	87,5%
Skenario 2	25	25	-	25	6	25	-	25	6	83,33%	25	6	83,33%
Skenario 3	11	14	4	14	-	14	4	14	4	82,14%	13	4	88,46%

Sementara itu berdasarkan Tabel 5 algoritma usulan memberikan hasil yang lebih baik dengan penelitian sebelumnya yang telah dipublikasi. Pada skenario 1 yang menggunakan data Mitchell (1957) algoritma usulan menghasilkan 5 stasiun kerja dengan efisiensi sebesar 87,5%. Hasil berbeda diperoleh dari penelitian Rahadian (2010) menggunakan VND yang menghasilkan jumlah stasiun kerja yang sama namun efisiensi hanya 80,77%. Pada penelitian Hoffman (1992) yang menggunakan EUREKA serta Goncalves dan Raimundo (2002) menggunakan *Hybrid Genetic Algorithm* menghasilkan jumlah stasiun kerja yang sama dengan pengujian algoritma yaitu 5 stasiun kerja. Pada penelitian Scholl (1997) tidak menggunakan set data Mitchell (1957). Set data pada skenario 2 algoritma usulan memberikan hasil yang sama dengan penelitian sebelumnya, namun pada skenario 3 efisiensi pada pengujian algoritma terbukti lebih baik jika dibandingkan dengan hasil penelitian sebelumnya. Jika dilihat dari tabel hasil perbandingan dengan penelitian sebelumnya, hasil pengujian algoritma lebih baik disebabkan oleh waktu siklus yang turun sehingga menghasilkan efisiensi lintasan yang lebih baik dari penelitian sebelumnya.

5. KESIMPULAN

Beberapa kesimpulan dari hasil penelitian ini adalah:

1. Pengembangan algoritma yang diusulkan adalah algoritma *variable neighbourhood descent with fixed threshold* yang ditujukan untuk menyelesaikan masalah pada lintasan perakitan tunggaldengan kriteria minimisasi jumlah stasiun kerja.
2. Pengujian terhadap sejumlah skenario data menunjukan bahwa algoritma usulan dapat digunakan untuk menyelesaikan SALBP I

3. Algoritma usulan memungkinkan ruang penerimaan solusi yang lebih besar dibandingkan penelitian-penelitian sebelum karena mempertimbangkan semua solusi yang muncul meskipun bukan solusi terbaik selama solusi tersebut berada dalam ambang batas penerimaan.

DAFTAR PUSTAKA

Dueck, G. dan Scheuer, T. (1990). Threshold Accepting. A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing. *Journal of Computation Physics* 90, 161-175.

Elsayed, E. A., dan Boucher, T. O. (1994). *Analysis and Control of Production Systems*. Second Edition. Prentice Hall New Jersey.

Goncalves, J. F., dan De Almeida, J. R. (2002). A Hybrid Genetic Algorithm for Assembly Line Balancing. *Journal of Heuristics*, 8, 629-642.

Hansen P. dan Mladenovic. (1999). An Introduction to Variable Neigbourhood Search. In S. Voss, S. Martello, L. H. Osman, and C. Roucairol, editors, *Meta Heuristics, Kluwer Academic Publisher*.

Hoffman, T.R. (1993). EUREKA: A Hybrid System for Assembly Line Balancing. *Management Science* 38, 39-7.

Rahadian, D. (2010). Model Keseimbangan Lintasan Perakitan Menggunakan Algoritma *Variable Neighbourhood Descent* Minimisasi Jumlah Stasiun Kerja. ITENAS.

Scholl, A. (1999). *Balancing and Sequencing of Assembly Lines*. Second Edition. Physica-Verlag Heidelberg New York.

Scholl, A. dan R. Klein. (1997). "SALOME: A Bidirectional Branch and Bound Procedure for Assembly Line Balancing. "*INFORMS Journal on Computing* 9, 319-334.