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

TIKA AGUSTIN, EMSOSFI ZAINI, ARIF IMRAN

Jurusan Teknik Industri
Institut Teknologi Nasional (Itenas) Bandung

Email: agustintika@yahoo.com

ABSTRAK

Keseimbangan lintasan perakitan merupakan masalah proses penempatan elemen kerja pada setiap stasiun kerja tanpa melanggar precendence constraint dan waktu siklus. Pada permasalahan ini lintasan yang diteliti yaitu lintasan perakitan paralel. Metode yang digunakan dalam Tugas Akhir ini adalah VND with fixed threshold. Tujuan dilakukannya penelitian ini yaitu untuk mengembangkan algoritma keseimbangan lintasan perakitan paralel menggunakan metode VND with fixed threshold dengan kriteria minimisasi jumlah stasiun kerja. Langkah yang dilakukan pada penelitian ini yaitu menggunakan perhitungan rank position weight (RPW) untuk initial solution. Pada tahap 2 yaitu local search terdiri dari proses exchange dan insert agar dapat meminimasi jumlah stasiun kerja. Algoritma usulan yang diuji menggunakan 3 skenario. Skenario 1 bertujuan untuk menguji cara kerja algoritma usulan, Skenario 2 dan Skenario 3 bertujuan untuk menguji keandalan algoritma usulan. Pada Skenario 1 mendapatkan hasil yang sama dengan literatur, pada Skenario 2 mendapatkan hasil yang lebih baik dari literatur, sedangkan pada Skenario 3 mendapatkan hasil yang tidak lebih baik dari literatur.

***Kata kunci**: keseimbangan lintasan, paralel, VND with fixed threshold*

ABSTRACT

The balance of the line assembly is a matter of the work placement element at each work station without breaking precendence constraint and cycle time. In this issue examined the line path parallel assembly. The method used in the final project is VND with fixed threshold. The purpose of this research is to develop algorithms balance trajectory parallel assembly VND method with fixed threshold criteria minimization of the number of work stations. Steps taken in this research is to use the calculation rank position weight (RPW) for the initial solution. In phase 2 which consists of local search and insert exchange process in order to minimize the number of work stations. Proposed algorithm is tested using three scenarios. Scenario 1 aims to examine how the proposed algorithm, Scenario 2

* *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*

and Scenario 3 aims to test the reliability of the proposed algorithm. In Scenario 1 get the same results with the literature, in Scenario 2 get better results from the literature, whereas in Scenario 3 to get better results from the literature.

***Keywords**: line balancing, parallel, VND with fixed threshold*

1. PENDAHULUAN

Keseimbangan lintasan perakitan (*Assembly Line Balancing*) merupakan masalah proses penempatan elemen kerja pada setiap stasiun kerja tanpa melanggar *precendenceconstraint* dan waktu siklus.Keseimbangan lintas perakitan berhubungan erat dengan produksi massal.Sejumlah pekerjaan perakitan dikelompokkan ke dalam beberapa pusat kerja.Tujuan akhir dari keseimbangan lintas adalah minimisasi waktu menganggur di tiap stasiun kerja. Pembuatan suatu produk umumnya dilakukan melalui beberapa tahapan proses produksi pada beberapa departemen yang berupa aliran proses produksi. Apabila terjadi hambatan atau ketidakefisienan dalam suatu departemen, akan mengakibatkan terjadinya waktu menunggu dan penumpukan material.

Scholl (1999) memaparkan *assembly line balancing* sebagai *simple assembly line balancing problem* I (SALBP I), *simple assembly line balancing problem* II (SALBP II), dan *simple assembly line balancing problem*-E (SALBP-E). Pada SALBP I, sejumlah elemen kerja dialokasikan pada beberapa stasiun kerja untuk minimisasi jumlah stasiun kerja, pada SALBP II minimisasi waktu siklus pada sejumlah elemen kerja, dan pada SALBP-E maksimisasi efisiensi lintasan.

Industri manufaktur biasanya mempunyai satu lintasan perakitan atau lebih.Jika permintaan banyak dan lintasan tunggal tidak mencukupi, maka dilakukan penggandaan lintasan perakitan. Keuntungan dari lintasan perakitan paralel dapat terlihat ketika ada stasiun kerja yang bermasalah, lintasan lainnya akan tetap berjalan. Sedangkan pada lintasan tunggal, lintasan tersebut harus dimatikan jika terdapat masalah dalam stasiun kerja.

SALBP telah dikembangkan salah satunya oleh Andres *et al.*(2008) yang menggunakan GRASP pada penelitiannya. Penelitian lain mengenai SALBP yang pernah dilakukan yaitu oleh Rahadian (2010) yang menggunakan VND dalam penelitiannya.

Menurut Suer dan Dagli (1994) keseimbangan lintasan perakitan paralel membutuhkan perhitungan heuristik dan algoritma untuk mendapatkan jumlah dari lintasan.Suer (1998) juga mempelajari alternatif strategi untuk keseimbangan lintasan perakitan paralel untuk produk tunggal.Terdapat beberapa penelitian yang telah mempelajari PALBP, yaitu salah satunya pada studi literatur Gocken *et al*. (2006) yang mengajukan model matematis untuk menyelesaikan PALBP dan mengusulkan dua algoritma heuristik. PALBP pernah diselesaikan oleh beberapa penelitian yang dapat dilihat dalam Tugas Akhir Purwaman (2011) antara lain Sudana (2011) dengan menggunakan GRASP, dan dilanjutkan oleh Purwaman (2011) dengan menggunakan algoritma VND.

Thresholdaccepting adalah metode metaheuristik untuk pencarian *local optimum* melalui bilangan random *feasible* yang dikembangkan oleh Dueck dan Scheuer (1990).Metode inidikembangkan untuk menyelesaikan masalah optimisasi kombinatorial.Imran dan

Okdinawati (2010) mengembangkan *threshold accepting* untk permasalahan *Heterogeneous Fleet Vehicle Routing Problem* (HFVRP).*Threshold accepting* menerima solusi yang masih dalam ambang batas dari solusi awal, sehingga menemukan solusi yang lebih baik. Berdasarkan literatur, penggabngan algoritma VND *with fixed threshold* belum dikembangkan, sehingga perlu dilakukan penelitian pengembangan algoritma VND *with fixed threshold.*

2. METODOLOGI PENELITIAN

Metodologi penelitian disusun secara sistematis dan terarah yang digunakan sebagai suatu kerangka dalam sebuah penelitian ilmiah.Adapun tahapan dalam sebuah penelitian ini adalah sebagai berikut.

2.1 Tahap Studi Literatur

Pada tahap ini dilakukan studi literatur mengenai keseimbangan lintasan perakitan lurus (*simple assembly line balancing problem*, SALBP) dan lintasan keseimbangan paralel (*parallel assembly line balancing problem*, PALBP) dengan algoritma VND *with fixed threshold.*

1. Beberapa penelitian sebelumnya mengusulkan pendekatan dengan menggunakan metaheuristik dalam menyelesaikan SALBP diantaranya Goncalves dan Almeida (2002) yang mengusulkan algoritma genetik campuran untuk menyelesaikan SALBP I dengan kriteria minimisasi jumlah stasiun kerja
2. Hansen dan Mladenovic (1999) mengembangkan VND pertama kali
3. Dueck dan Scheuermemperkenalkan *threshold accepting* pertama kali

2.2 Tahap Identifikasi Posisi Penelitian

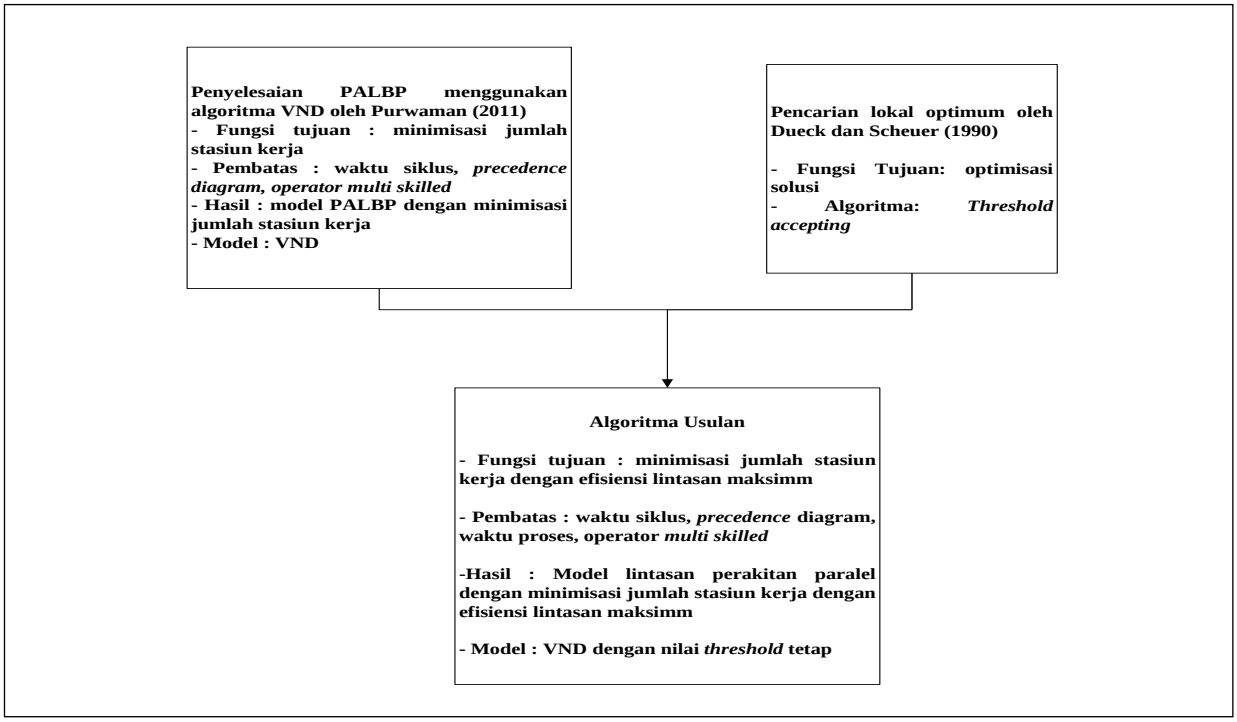
Berdasarkan studi literatur, belum pernah dilakukan pengembangan algoritma VND *with fixed threshold* untuk PALBP. Untuk itu perlu dilakukan penelitian penerapan VND *with fixed threshold* untuk menyelesaikan permasalahan PALBP dengan kriteria minimisasi jumlah stasiun kerja. Peta posisi penelitian-penelitian yang telah dilakukan dapat dilihat pada Gambar 1.

				Metaheuristik		
		Optimasi	Heuristik	GRASP	VND	VND *with Fixed Threshold*
Keseimbangan Lintasan Perakitan	SALBP	Elsayed dan Boucher (1994)	Talbot *et al* . (1989) Scholl dan Vo (1996) Ponnambalan et al. (1999)	Andres *et al* . (2008)	Rahadian (2010)	
	PALBP	Scholl dan Boysen (2008)	Suer dan Dagli (1994) Gocken *et al* . (2006) Suer (1998)	Sudana (2011)	Purwaman (2011)	Penelitian

Gambar 1. Peta Posisi Penelitian

2.3 Tahap Pengembangan Algoritma

Pengembangan algoritma yang dilakukan dalam penelitian ini menggunakan VND *with fixed threshold*untuk menyelesaikan keseimbangan lintasan perakitan paralel dengan kriteria minimisasi jumlah stasiun kerja dengan efisiensi lintasan maksimum. Penelitian yang dilakukan merupakan pengembangan algoritma dari penelitian sebelumnya, skema pengembangan algoritma dapat dilihat pada Gambar 2.



Gambar 2.Skema Pengembagan Algoritma

2.4 Tahap Pengujian Algoritma dan Analisis

Skenario yang digunakan dalam penelitian ini dengan pengujian algoritma sebagai berikut:

1. Skenario 1 yang bertujuan untuk menguji cara kerja algoritma usulan dengan menggunakan data set dari penelitian Merten (1967) dengan jumlah elemen kerja n=7
2. Skenario 2 bertujuan untuk menguji keandalan algoritma usulan dengan menggunakan set data Jaeschke (1964) dengan jumlah elemen kerja n = 9
3. Skenario 3 dengan menggunakan set data Scholl and Boysen (2008) dengan jumlah elemen kerja n = 18
   Hasil dari algoritma usulan kemudian dibandingkan dengan hasil penelitian yang sudah dipublikasikan.

2.5 Kesimpulan dan Saran

Sub bab ini berisi mengenai kesimpulan dari penelitian ini dan saran-saran untuk penelitian selanjutnya.

3. PENGUMPULAN DAN PENGOLAHAN DATA

Pengembangan algoritma pada penelitian ini dapat dijelaskan sebagai berikut:

1. Tahap 1 Pembentukan Solusi Inisial
   Tahap ini dimana tahap pembentukan solusi inisial dengan menggunakan algoritma *rank positional weight* (RPW) dengan memperhatikan urutan *positional weight* terbesar.
2. Tahap 2 *Local Search*
   Setelah mendapatkan hasil dari pembentukan inisial solusi, selanjutnya dilakukan pencarian solusi pada *neighborhood* yang terbentuk dari sekumpulan elemen kerja dengan melakukan modifikasi pada *neighborhood.*Tujuannya adalah untuk mendapatkan efisiensi yang lebih besar dengan nilai *smoothness index* yang lebih kecil, kemudian dibandingkan hasilnya dengan inisial solusi. Perubahan struktur pada *neighborhood* ini dibagi menjadi dua tahap yaitu:

a. *Exchange*
   Teknik *Exchange* merupakan perubahan struktur dengan cara melakukan pertukaran posisi dari dua elemen kerja yang terpilih pada dua stasiun kerja yang berbeda.
b. *Insert*
   Teknik *insert* merupakan perubahan struktur dengan cara menyisipkan sebuah operasi kedalam operasi lainnya.

Notasi-notasi yang digunakan dalam algoritma ini adalah sebagai berikut:
i : indeks untuk stasiun kerja (SK); (i = 1, 2, ..., m)
j : indeks untuk elemen kerja; (j = 1, 2, ..., n)
t_j : waktu proses pada elemen kerja ke-j
CT : waktu siklus
ST_i : akumulasi waktu elemen kerja pada stasiun kerja ke-i
EL : efisiensi lintasan
SI : *smoothness index*
IT_i : *idle time* pada stasiun kerja ke-i
IT_{maks} : *idle time* dengan nilai terbesar
NI_i : *neighborhood structure* yang di*insert* pada stasiun kerja ke-i
NX_i : *neighborhood structure* yang ditukar pada stasiun kerja ke-i
NY_i : *neighborhood structure* penukar pada stasiun kerja ke-i
VX_j : elemen kerja yang ditukar
VY_j : elemen kerja penukar
e : indeks iterasi untuk proses *exchange* (e= 1, 2, ..., e_{maks})
EL_{Th} :minimum efisiensi dari solusi yang dapat diterima (EL_{Th}= EL_{terbaik}* $\%T_h$)
$\%T_h$: persentase batas *threshold*

Langkah-langkah algoritma VND *with fixed threshold* untuk menyelesaikan PALBP dengan minimisasi jumlah stasiun kerja dengan efisiensi lintasan maksimum:

TAHAP 1 *INITIAL SOLUTION*
Langkah 1
Inputkan j, t_j, CT dan *precedence diagram.*
Langkah 2
Buatlah solusi inisial dari permasalahan yang ada dengan menggunakan metoda *Rank Positional Weight* (RPW) dan simpanlah solusi tersebut sebagai *current solution.*
Berikut langkah-langkah initial solution menggunakan metode RPW:
2.1 Buat diagram *precedence diagram* dari proses yang bersangkutan.
2.2 Hitung *positional weight* (PW) untuk setiap elemen kerja. PW untuk suatu elemen berhubungan dengan waktu terpanjang dari awal operasi sampai dengan akhir operasi.
2.3 Urutkan elemen kerja berdasarkan *positional weight*. Elemen dengan PW peringkat yang tinggi ditempatkan terlebih dahulu.
2.4 Tempatkan elemen kerja ke dalam stasiun kerja. Elemen kerja yang memiliki PW dan peringkat yang tinggi ditempatkan terlebih dahulu.
2.5 Apabila setelah satu elemen kerja ditempatkan ke dalam stasiun kerja masih terdapat sisa waktu, tempatkan elemen kerja yang memiliki peringkat tertinggi berikutnya selama tidak menyalahi *precedencediagram*.
2.6 Ulangi langkah 2.4 dan 2.5 sampai seluruh elemen kerja sudah ditempatkan ke dalam stasiun kerja.

TAHAP 2 *LOCAL SEARCH*

Langkah 3

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

Langkah 4

Set solusi inisial sebagai *current solution*

Langkah 5

Set k=1, k_{maks}=[n/2|

Langkah 6

Set e = 1 dan e_{maks} = m

Langkah 7

Tentukan *idle time*(IT)disetiap stasiun kerja dengan persamaan $IT = \{CT\text{-}ST_i\}$

Langkah 8

Pilih nilai ITiyang terbesar (IT_{maks}) di setiap stasiun kerja dengan persamaan $IT_{maks} = \underset{\forall i}{^{maks}}\{ITi\}$ dan set sebagai NX_i. Jika terdapat lebih dari 1 stasiun kerja yang memiliki nilai IT_{maks}sama, maka pilihlah stasiun kerja secara random.

Langkah 9

Periksa apakah $|j \in NX_i| = 1$?

Jika ya, set j yang terpilih sebagai VX_jdan lanjutkan ke Langkah 11.

Jika tidak, lanjutkan ke Langkah 10.

Langkah 10

Pilih VX_j, $j \in NX_i$menggunakan persamaan $VX_j = arg^{maks}_{j \in NXi} \{tj\}$ kemudian set j dengan nilai terbesar sebagai VX_j. Jika terdapat nilai $t_j$yang sama, maka pilihlah secara random.

Langkah 11

Pilih VY_j, $j \notin NX_i$ dan lanjutkan ke Langkah 11.1

Langkah 11.1

Apakah proses *exchange* melanggar *precedence constrain* atau VY_j sudah pernah menghasilkan konfigurasi yang sama ?

Jika ya, lanjutkan ke Langkah 11.2.Jika tidak, lanjutkan ke Langkah 11.3

Langkah 11.2

Apakah masih terdapat $j \notin NX_i$ yang belum ditukar ?

Jika ya, kembali ke Langkah 10.Jika tidak, set $e = e + 1$, dan lanjut ke Langkah 12.

Langkah 11.3

Lakukan proses *exchange* untuk semua j yang memungkinkan. Hitunglah *EL*&*SI* di setiap lintasan.Periksa apakah $EL \geq EL_{Th}$?

Jika ya, lanjut ke Langkah 11.4.Jika tidak, kembali ke Langkah 11.2

Langkah 11.4

Apakah $|VY_j$ dalam batas $EL \geq EL_{Th}|>1$?

Jika ya, pilih j dengan nilai *EL* terbesar, jika terdapat *EL* yang sama pilih nilai *SI* terkecil.

Jika terdapat nilai *SI* yang sama pilihlah secara random dan lanjutkan ke Langkah 11.5

Jika tidak, lanjut ke Langkah 11.5

Langkah 11.5

Apakah *EL*=100% ?Jika ya, lanjut ke Langkah 17.Jika tidak, lanjut ke Langkah 13

Langkah 12

Apakah $e > e_{\text{maks}}$?

Jika ya, lanjut ke Langkah 13

Jika tidak, kembali dan ulangi Langkah 7.

Langkah 13

Set *neighborhood structure insert* (NI_i) = 1 dan $i_{maks} = m$

Langkah 14

Pilih $j \notin NI_i$yang akan di *insert* ke NI_idan lanjutkan ke Langkah 14.1

Langkah 14.1

Apakah terdapat *j* yang dapat dilakukan proses *insert* melanggar *precedence constrain* terhadap NI_i atau apakah $ST_i > CT$?
Jika ya, lanjutkan ke Langkah 14.3
Jika tidak, set $NI_i = NI_{i+1}$, dan lanjutkan ke Langkah 14.2

Langkah 14.2
Apakah *j* yang di *insert* terhadap NI_i sudah pernah menghasilkan konfigurasi yang sama ?
Jika ya, set $NI_i = NI_{i+1}$, dan kembali ke Langkah 14.3
Jika tidak, lanjut ke Langkah 14.4

Langkah 14.3
Apakah $NI_i = NI_{m+1}$?
Jika ya, lanjut ke langkah 17
Jika tidak, kembali dan ulangi Langkah 14

Langkah 14.4
Lakukan proses *insert* untuk semua *j* yang memungkinkan. Apakah jumlah SK berkurang?
Jika ya, lanjut ke Langkah 14.7
Jika tidak, lanjut ke Langkah 14.5

Langkah 14.5
Hitung nilai *EL* dan *SI*.Apakah $EL \geq EL_{Th}$?
Jika ya, lanjutkan ke Langkah 14.6
Jika tidak, set $NIi = i + 1$, kembali ke Langkah 14.3

Langkah 14.6
Apakah terdapat |*j* hasil *insert* dalam batas $EL \geq EL_{Th}$ | >1 ?
Jika ya, pilih *j* dengan nilai *EL* terbaik. Jika terdapat nilai *EL* yang sama maka pilihlah nilai *SI* yang terbaik. Jika sama pilihlah secara random. Lanjutkan ke Langkah 14.7
Jika tidak, ke Langkah 14.7

Langkah 14.7
Apakah *EL*=100% ?
Jika ya, ke Langkah 17
Jika tidak, ke Langkah 15

Langkah 15
Pilihlah nilai *EL* terbaik dari *Current Solution*, *Exchange, Insert* dan set konfigurasi *EL* terbaik sebagai *current* solusi untuk iterasi selanjutnya. Set $k=k + 1$

Langkah 16
Apakah $k > [n/2]$?
Jika ya, lanjut ke Langkah 17
Jika tidak, kembali dan ulangi Langkah 6

Langkah 17
Simpan dan tampilkan konfigurasi terbaik

4. PENGUJIAN DAN ANALISIS

4.1 Pengujian Algoritma

Pengujian algoritma VND *with Fixed Threshold* dilakukan dengan tiga skenario, yaitu skenario 1 menguji cara kerja dari algoritma usulan dengan set data Merten (1967) dalam Gocken *et al.* (2006), sedangkan skenario 2 digunakan untuk menguji nilai *threshold* dari set data Jaeschke (1964) dalam Gocken *et al.* (2006) dan skenario 3 menggunakan set data Scholl dan Boysen (2008). Setiap set data akan diujikan dengan 3 buah nilai *threshold* yaitu 5%, 10%, 15%. Berikut ini adalah set data yang digunakan pada tahap pengujian algoritma.

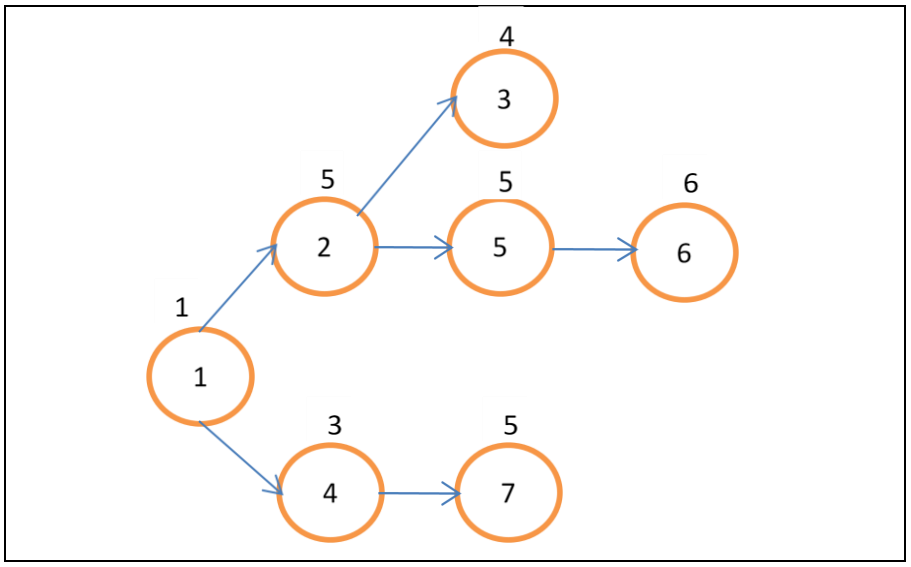
1. Skenario 1

Skenario 1 digunakan untuk menguji cara kerja algoritma dengan menggunakan set data Merten (1967) dalam Gocken *et al.* (2006). Dari data yang digunakan, data ini

menggunakan produk yang sama, data dengan jumlah elemen kerja (*n*) = 7 dan waktu siklus (*CT*) = 9. Set data untuk skenario 1 dan *precedence diagram* dapat dilihat pada Tabel 1 dan Gambar 3.

Tabel 1. Set Data Skenario 1

j	t_j	*Successor*
1	1	2,4
2	5	3,5
3	4	-
4	3	7
5	5	6
6	6	-
7	5	-



Gambar 3.*Precedence Diagram* Set Data Skenario 1

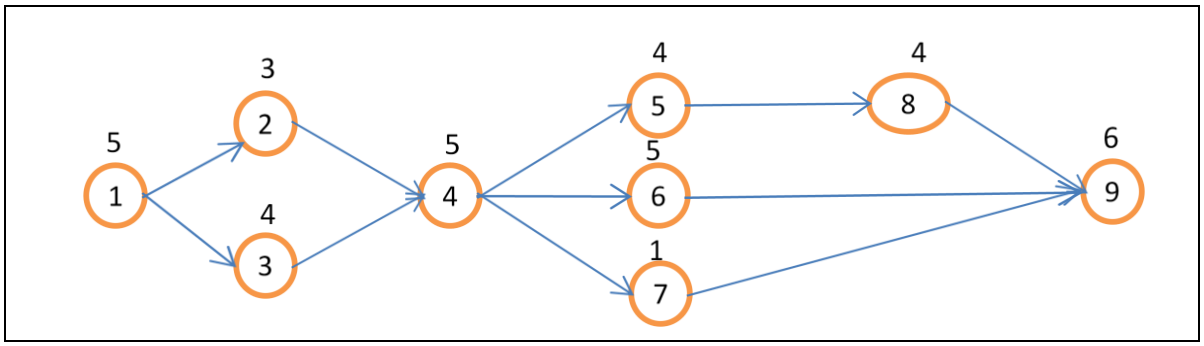
2. Skenario 2

Set data yang digunakan pada skenario 2 adalah set data Jaeschke (1964) dalam Gocken *et al.* (2006) yang memiliki jumlah elemen kerja (*n*) = 9 dengan waktu siklus (*CT*)= 11. Data waktu proses dapat dilihat pada Tabel 2.

Tabel 2. Set Data Skenario 2

j	t_j	*Successor*
1	5	2,3
2	3	4
3	4	4
4	5	5,6,7
5	4	8
6	5	9
7	1	9
8	4	9
9	6	-

Precedence diagram untuk set data skenario 2 dapat dilihat pada Gambar 4.



Gambar 4.*Precedence Diagram* Set Data Skenario 2

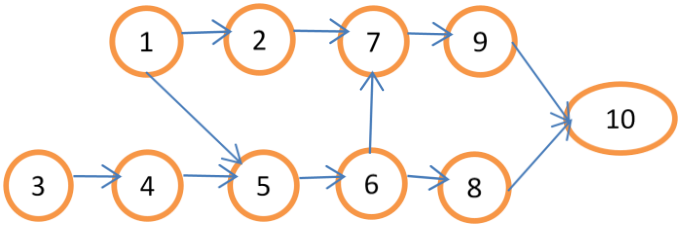
3. Skenario 3

Skenario 3 menggunakan set data Scholl dan Boysen (2008) yang memiliki jumlah elemen kerja (n) = 18 dengan waktu siklus (CT) = 10. Data waktu proses (t_j) dapat dilihat pada Tabel 3.

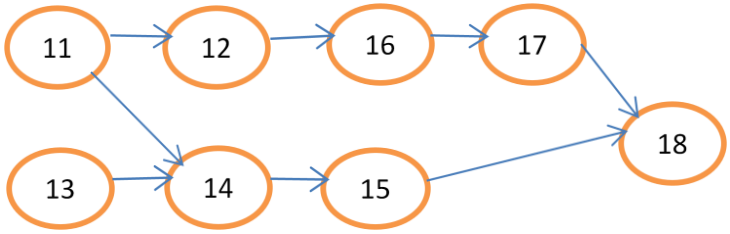
Tabel 3. Set Data Skenario 3

j	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
t_j	5	6	7	5	5	5	4	2	5	2	4	4	3	1	5	4	5	8

Precedence diagram untuk set data skenario 3 dapat dilihat pada Gambar 5.



a. Produk 1



b. Produk 2

Gambar 5.*Precedence Diagram* Set Data Skenario 3

Hasil pengujian algoritma pada setiap skenario untuk masing-masing nilai *threshold* dapat dilihat pada Tabel 4

Tabel 4. Hasil Pengujian Algoritma

Set Data	Hasil Pengujian Algoritma					
	15%		10%		5%	
	EL	*SI*	*EL*	*SI*	*EL*	*SI*
Merten (1967)	71,60%	8,89	71,60%	8,89	71,60%	8,89
Jaeschke (1964)	84.09%	6.00	84.09%	6.00	84.09%	6.00
Scholl dan Boysen (2008)	80%	7,35	80%	7,35	80%	7,35

Perbandingan hasil pengujian algoritma usulan dengan hasil penelitian-penelitian sebelumnya dapat dilihat pada Tabel 5.

Tabel 5. Tabel Hasil Perbandingan Algoritma Usulan dengan Penelitian Sebelumnya

Set Data	Optimasi	Heuristik	Metaheristik	Jumlah Elemen Kerja (*n*)	Waktu Siklus (*CT*)	Jumlah Stasiun Kerja	*EL* (%)	*SI*
Merten (1967)		Gocken *et al.* (2006)		7	9	9	71,60%	
			Purwaman (2011)	7	9	9	71,60%	8,54
			Algoritma usulan	7	9	9	71,60%	8,89
Jaeschke (1964)		Gocken *et al.* (2006)		9	11	8	74,74%	
			Purwaman (2011)	9	11	8	84,09%	3,46
			Algoritma usulan	9	11	8	84,09%	6.00
Scholl dan Boysen (2008)	Scholl dan Boysen (2008)			18	10	9	80%	
			Purwaman (2011)	18	10	9	88,89%	4,12
			Algoritma usulan	18	10	9	80%	7,35

4.2 Analisis

Berdasarkan Tabel 4 terdapat 3 skenario data yang diujikan dan masing-masing diujikn pada 3 nilai *fixed threshold* berbeda*.* Tidak terdapat perbedaan nilai *fixed threshold* pada skenario 1, skenario 2, dan skenario 3. Perbedaan hanya pada jumlah iterasi yang dilakukan pada masing-masing nilai *fixed threshold*. Semakin besar nilai *fixed threshold* maka semakin besar besar nilai ambang batas yang diijinkan untuk menerima solusi baru. Begitu pula sebaliknya, semakin kecil nilai *fixed threshold* maka semakin kecil nilai ambang batas yang diijinkan untuk menerima solusi baru.

Penelitian ini dilakukan dengan menggunakan algoritma VND dengan nilai *threshold* tetap.Penyelesaian masalah algoritma VND dengan nilai *threshold* tetap sama dengan penelitian sebelumnya yaitu dengan perubahan struktur *neighborhood exchange* dan *insert*, hanya terdapat perbedaan pada penerimaan solusi. Pada Tabel 5 ditampilkan perbandingan hasil algoritma usulan dengan beberapa penelitian sebelumnya.Dapat dilihat untuk set data skenario 1 telah digunakan oleh penelitian Gocken *et al.* (2006) dan penelitian lain yaitu Purwaman (2011) yang menghasilkan 9 SK dengan *CT* sebesar 9 dan efisiensi lintasan sebesar 71,62%. Algoritma usulan mendapatkan hasil yang sama yaitu sebanyak 9 SK dengan *CT* sebesar 9 dan efisiensi lintasan sebesar 71,62%. Pada set data skenario 2, penelitian yang dilakukan sebelumnya mendapatkan hasil yaitu dengan 8 SK dan *CT* =11 serta efisiensi lintasan 74,74%. Penelitian Purwaman (2011) dan algoritma usulan mendapatkan konfigurasi akhir yang sama yaitu dengan 8 SK dan *CT* = 11 serta efisiensi sebesar 84,09%. Pada set data skenario 3, penelitian sebelumnya dan algoritma usulan menghasilkan hasil yang sama yaitu dengan jmlah SK 9 dan *CT*= 10 serta efisiensi 80%. Hasil ini tidak lebih baik dari penelitian Purwaman (2011) yang menghasilkan efisiensi lintasan sebesar 88,89% dengan 9 stasiun kerja dan *CT*=10.

5. KESIMPULAN

Kesimpulan dari penelitian yang telah dilakukan adalah:

1. Algoritma yang digunakan dalam merancang model keseimbangan lintasan perakitan paralel yaitu *variable neighborhood descent* (VND)*with fixed threshold.*

2. Algoritma keseimbangan lintasan perakitan paralel yang dikembangkan adalah algoritma menggunakan lintasan perakitan paralel dengan kriteria minimisasi jumlah stasiun kerja dengan efisiensi lintasan maksimum .
3. Pada pengujian algoritma menggunakan set data Jaeschke (1964) menghasilkan efisiensi lintasan yang lebih baik daripada efisiensi lintasan pada penelitian sebelumnya.
4. Pada set data Scholl dan Boysen (2008) didapat hasil akhir dari algoritma usulan tidak lebih baik dari penelitian sebelumnya.
   Pada set data Merten didapat hasil akhir algoritma usulan yang menghasilkan nilai yang sama dengan penelitian sebelumnya.

DAFTAR PUSTAKA

Andrés, C., Miralles, C., dan Pastor, R. (2008). Balancing and Scheduling Tasks in Assembly Lines with Sequence-dependent Setup Times.*European Journal of Operational Research,* 187, 1212-1223.

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

Gocken, H., Agpak, K., dan Benzer, R. (2006). Balancing of parallel assembly lines.*Int. J. Production Economics,* 103, 600-609.

Imran, A., dan Okdinawati, L. (2010). A Threshold Accepting Heuristik for The Heterogeneous Fleet Vehicle Routing Problem *Jurnal of business logistics,* 15-26.

Ponnambalam, S. G., Aravindan, P. dan Naidu, G. M. (1999). A Comparative Evaluation of Assembly Line Balancing Heuristics. *International Journal of Advanced Manufacturing Technology* 15:577–586.

Purwaman, G. A. (2011). *Model Keseimbagan Lintasan Perakitan Paralel Menggunakan AlgoritmaVariable Neighborhood Descent dengan Kriteria Minimisasi Jumlah Stasiun Kerja.*ITENAS.

Rahadian, D. (2010). *Model Keseimbangan Lintasan Perakitan Menggunakan Algoritma variable Neighborhood 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 N. Boysen. (2008). Designing parallel assembly lines with split workplaces: Model and optimization procedure. *International Journal of Production Economics.*

Suer, G.A. (1998). Designing parallel assembly lines.*Computers and Industrial Engineering,*35, 467-470.

Suer, G., dan Dagli, C. (1994). A Knowledge-Based System for Selection of Resource Allocation Rules and Algorithms. In: Mital, A., Anand, S. (Eds), *Handbook of Expert System Applications in Manufacturing; Structures and Rules. Chapman & Hall,* London, 108-147.

Talbot, F. B., Gehrlein, W. V. dan Patterson, J. H. (1986). Comparative Evaluation of Heuristics Line Balancing Techniques. *Management Science*, 32, 430-454.