ALGORITMA *VARIABLE NEIGHBORHOOD DESCENT WITH FIXED THRESHOLD* UNTUK KESEIMBANGAN LINTASAN PERAKITAN *TWO SIDED* DENGAN KRITERIA MINIMISASI JUMLAH STASIUN KERJA*

INAYAH, ALEX SHALEH, EMSOSFI ZAINI

Jurusan Teknik Industri
Institute Teknologi Nasional (Itenas) Bandung

Email: nayahbaisa@gmail.com

ABSTRAK

Lintasan two-sided adalah dua lintasan seri (bagian sisi kiri dan bagian sisi kanan) yang bekerja secara pararel. Pekerjaan harus dikerjakan pada bagian kiri atau pada bagian kanan dari sisi perakitan, namun ada juga pekerjaan yang dapat dikerjakan dari kedua sisinya. Algoritma Variable Neighborhood Descentwith Fixed Threshold (VND with Fixed Threshold) memungkinkan penerimaan ruang solusi lebih besar karena mempertimbangkan semua solusi yang muncul meskipun bukan solusi yang lebih baik. VND with Fixed Threshold memanfaatkan perubahan struktur yang terjadi dalam neighborhood untuk mengatasi masalah jika pencarian solusi terjebak dalam minimum lokal. VND with Fixed Threshold terdiri atas tahap pembangkitan solusi inisial menggunakan algoritma region approach dan tahap perbaikan solusi inisial dalam pencarian local search (exchange dan insert).

***Kata kunci**: keseimbangan lintasan perakitan, lintasan perakitan two-sided, metaheuristik, varible neighborhood descent with fixed threshold.*

ABSTRACT

Two-sided line is a two serial line (on the left side and the right side) working togetherin parallel. Work to be done on the left or on the right side of the assembly line, and there is also work that can be done from both sides. Variable neighborhood descent algorithm with fixed threshold (VND with Fixed Threshold) allows a larger solution space of acceptance because it considers all the solutions that emerge even though not the better solution. VND with Fixed Threshold utilize structural changes that occur in the neighborhood to address issues if the search was stuck in a local minimum solution. VND with Fixed Threshold stages consist of generating the initial solution using region approach algorithm and then improving the initial solution using local search (exchange and insert).

***Kata kunci**: assembly line balancing, two-sided assembly line balancing, metaheuristic, varible 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

Lintasan perakitan (*assembly line*) merupakan suatu lintasan produksi berupa lintasan materialatau *material in process* yang mengalami beberapa proses perakitan, mulai dari proses rakitan awal sampai menjadi sebuah produk akhir (Scholl, 1999).Pada lintasan perakitan muncul masalah bagaimana lintasan perakitan tersebut menjadi suatu lintasan yang seimbang, agar perusahaan dapat memperoleh efisiensi lintasan yang paling tinggi.Keseimbangan lintasan perakitan (*Assembly Line Balancing*, ALB)merupakan suatu proses penempatan elemen kerja pada setiap stasiun kerja tanpa melanggar *precedence* yang ada dan waktu siklus.Scholl (1999) mengklasifikasikan permasalahan ALB menjadi permasalahan keseimbangan lintasan perakitan *simple assembly line balancing problem type* I (SALBP I), SALBP II, SALBP-E dan SALBP-F. SALBP I merupakan permasalahan ALB yang menempatkan sejumlah elemen kerja pada beberapa stasiun kerja sehingga jumlah stasiun kerja dapat diminimisasi. SALBP II merupakan pengalokasian sejumlah elemen kerja pada stasiun kerja dengan jumlah stasiun kerja yang telah diketahui sebelumnya sehingga waktu siklus dapat diminimisasi.

Lintasan perakitan tunggal (*single line)* dinilai menjadi tidak efisien lagi ketika dihadapkan pada perakitan produk dengan kompleksitas tinggi.Untuk mengatasi masalah tersebut, dikembangkan lintasan perakitan dua sisi (*two-sided assembly line balancing,* TALB).Pekerjaan TALBuntuk kedua sisinya dikerjakan secara bersamaan. Pekerjaan yang dilakukan pada sisi kanan dan kiri ketika memiliki perbedaan waktu yang signifikan mengakibatkan ketidakseimbangan lintasan sehingga efisiensi lintasan menjadi kecil. Masalah keseimbangan lintasan perakitan *two-sided* (*two-sided assembly line balancing problem,* TALBP) pertama kali dipelajari oleh Bartoholdi (1993) dengan menggambarkan desain dan penggunaan sebuah program komputer interaktif berbasis aturan prioritas sederhana untuk lintasan perakitan dua sisi, contoh produknya truk.TALBP merupakan permasalahan optimasi kombinatorial (*non-deterministic polynomial hard* (NP-*hard*)) sehingga struktur kombinatorial TALBP memunculkan kesulitan untuk mendapatkan solusi dalam waktu singkat dengan metode optimasi.

Algoritma metaheuristik *Variable Neighborhood Descent* (VND) adalah algoritma untuk memecahkan masalah dengan menggunakan set *neighborhood* dalam pencarian dan penemuan *local search* sehingga akan mencapai global optimum yang dikembangkan oleh Hansen dan Mladenoviv,1999. Dalam proses pencarian global optimum, VND melakukan pergantian set *neighborhood* secara deterministik. Apnena (2011) menyelesaikan masalah TALBP dengan algoritma VND dengan kriteria minimisasi jumlah stasiun kerja.

Metode metaheuristik lainnya adalah *threshold accepting* yang dikembangkan oleh Dueck dan Scheuer (1990). *Threshold accepting*adalah metode metaheuristik untuk pencarian solusi optimum kombinatorial yang memiliki nilai ambang batas penerimaan solusi. Semua solusi yang muncul selama lebih baik, menghasilkan ongkos lebih murah atau sama dengan nilai variable *threshold*akan diterima.

Nilai ambang batas threshold dapat diadaptasi atau digunakan bersamaan dengan algoritma lain, sehingga dalam penelitian kali ini, nilai threshold akan digunakan bersamaan dengan algoritma VND. Penelitian-penelitian sebelumnya, tidak ada yang mempertimbangkan solusi yang bukan solusi terbaik, sehingga bisa dikembangkan algoritma yang mempertimbangkan solusi yang muncul walaupun bukan solusi terbaik yang diharapkan dapat memperbesar

ruang solusi penerimaan. Algoritma Variable Neighborhood Descent with Fixed Threshold (VND with FixedThreshold) memungkinkan penerimaan ruang solusi lebih besar karena mempertimbangkan semua solusi yang muncul meskipun bukan solusi yang lebih baik. Dalam pengembangan algoritma kali ini, algoritma *thresholdaccepting*digunakan untuk penentuan batas minimal penerimaan efisiensi lintasan dari solusi yang muncul. Ambang batasnya berupa nilai *fixedthreshold* yang semakin lama semakin mendekati 0.

Tujuan dari penelitian ini adalah menyelesaikan masalah pada lintasan perakitan *two sided* dengan menggunakan algoritma VND *with FixedThreshold* dengan kriteria minimisasi jumlah stasiun kerja.

2. METODOLOGI PENELITIAN

Metodologi penelitian yang dilakukan dalam penelitianini dijelaskan sebagai berikut.

2.1 Studi Literatur

Studi literatur berisi mengenai landasan teori yang berhubungan dengan *assembly line balancing, two-sided assembly line balancing, variable neighborhood descent* dan *threshold accepting.*

2.2 Identifikasi Posisi Penelitian

Peta posisi penelitian terhadap beberapa beberapa penelitian yang lalu dapat dilihat pada Tabel 1.

Tabel 1. Peta Posisi Penelitian

		Minimisasi jumlah stasiun kerja						
		Optimasi	Heuristik	Metaheuristik				
				Genetik Algoritm	GRASP	*Guided* GRASP	VND	VND *with Fixed* Th
ALB	SALBP	Hackman *et al.* (1989)	Baybars (1986), Talbot *et al.* (1986), Scholl dan Voβ (1996), Ponnambalan *et all.* (1999)	Andreson dan Ferris (1994), Goncalves dan de Almeida (2002)	Andres *et al.* (2008)		Rahardian (2010)	Hardini (2012)
	TALBP	Wu *et al.* (2008)	Lee *et al.* (2001)	Kim *et al.* (2000), Kim *et al* . (2009)	Pratama (2010)	Pradikta (2011)	Apnena (2011)	Penelitian

2.3 Pengembangan Algoritma

Penelitian ini merupakan pengembangan dari penelitian-penelitian sebelumnya yaitu:

1. Apnena (2011) yang mengembangkan metode *variable neighborhood descent*pada lintasan *two-sided*.
2. Dueck dan Scheuer (1990) mengembangkan metode metaheuristik untuk pencarian solusi optimum kombinatorial dengan algoritma *threshold accepting*.

Nilai ambang batas *threshold* dapat diadaptasi atau digunakan bersamaan dengan algoritma lain, sehingga dalam penelitian kali ini, nilai *threshold* digunakan bersamaan dengan algoritma VND. Tujuan penelitian untuk minimisasi jumlah stasiun kerja sehingga diperoleh efisiensi lintasan yang maksimum.

2.4 Pengujian Algoritma

Pengujian cara kerja model usulan dilakukan dengan menggunakan empat skenario, yaitu:

1. Skenario 1, skenario 2, dan skenario 4 menggunakan set data dari Kim *et al* (2000) dengan jumlah elemen pekerjaan (*j*) adalah 12, 12 dan 24 yang bertujuan untuk menguji cara kerja algoritma usulan.

2. Skenario 3 menggunakan set data dari Lee *et al* (2001) dengan jumlah elemen pekerjaan (*j*) 16.

Setiap skenario diujikan pada 3 nilai *threshold* yaitu 25%, 15%, 5%. Hasil pengujian dari keempat skenario tersebut kemudian dibandingkan dengan hasil dari beberapa penelitian sebelumnya.

2.5 Kesimpulan

Berisikan hal-hal penting yang diperoleh dari hasil penelitian.

3. PENGEMBANGAN ALGORITMA

Dalam pengembangan algoritma ini, notasi-notasi yang digunakan 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
EL_{Terbaik} = efisiensi lintasan terbaik
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, ..., *emaks*)
EL_{Th} = Batas penerimaan solusi efisiensi lintasan; $EL_{\text{Th}} = EL_{\text{Terbaik}} - (EL_{\text{Terbaik}} \times \% \; Th)$

Tahap-tahap yang dilakukan pada algoritma usulan adalah:

1. Tahap 1 Pembentukan Solusi Inisial
   Tahap 1 adalah tahap pembentukan solusi inisial dengan menggunakan metode Kilbridge-Wester atau *Region Approach* (RA). Proses pengerjaannya adalah dengan membagi *region* pada *precedence diagram*. Setiap *region* terdiri atas elemen kerja yang berada dalam satu kolom.
2. Tahap 2 *Local Search*
   Tahap ini adalah pencarian solusi yang lebih baik dengan cara mengubah struktur *neighborhood.* Terdapat 2 proses untuk mengubah struktur *neighborhood* pada algoritma usulan ini, yaitu:
   a. *Exchange*
      Exchange adalah salah satu cara untuk melakukan perubahan struktur *neighborhood*. Prosesnya adalah melakukan pertukaran 2 elemen kerja yang terpilih. Pertukaran dapat dilakukan selama tidak melanggar *precedence constraint,* dan pada lintasan *two-sided* tidak menyalahi posisi penempatan elemen kerja pada lintasan sisi kiri atau kanan.
   b. *Insert*
      Cara yang kedua untuk melakukan perubahan struktur *neighborhood* adalah dengan *insert*. Prosesnya adalah menyisipkan elemen kerja terpilih pada stasiun kerja lain

selama tidak melanggar *precedence constraint,* dan tidak melanggar posisi kanan, kiri dari elemen kerja.

Tahap 1 Pembentukan Solusi Inisial

Langkah 0

Inputkan *j, tj, CT* dan *precedence diagram.*

Langkah 1

Bentuk solusi inisial menggunakan metode Kilbridge-Wester (*Region Approach*) dengan prosedur sebagai berikut:

1.1 Elemen kerja pada *precedence diagram* dikelompokkan ke dalam kolom, semua elemen kerja yang tidak memiliki elemen kerja sebelumnya dikelompokkan pada kolom I. Elemen kerja setelah elemen kerja kolom I dikelompokkan pada kolom II. Lanjutkan proses ini dengan cara yang sama sampai semua elemen kerja telah dikelompokkan.

1.2 Tempatkan elemen kerja dari kolom ke I ke stasiun kerja dengan jumlah waktu elemen kerja tidak melebihi *CT*.

1.3 Hilangkan elemen yang telah ditempatkan dari total elemen kerja dan ulangi langkah 1.3.

1.4 Jika waktu stasiun melebihi *CT* karena memasukkan suatu elemen kerja tertentu, elemen kerja ini harus ditempatkan pada stasiun kerja berikutnya.

1.5 Ulangi langkah 1.3 sampai dengan 1.4 sehingga semua elemen ditempatkan pada stasiun kerja.

Langkah 2

Buatlah *layout* dari hasil perhitungan menggunakan metode RA kemudian hitung nilai efisiensi (*EL*) dan *smoothness index* (*SI*).

Tahap 2 *Local Search* dengan *VND with Fixed Threshold*

Langkah 3

Inputkan data awal berupa *precedencediagramj*, *tj*, *CT*, inputkan *EL*, *SI, ST*dari solusi inisial yang telah terbentuk, % *Th* dan EL_{Th}.

Langkah 4

Set *currentsolution* = solusi inisial.

Langkah 5

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

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 IT_iyang terbesar (IT_{maks}) dari seluruh stasiun kerja dengan persamaan sebagai berikut: $IT_{maks} = \overset{maks}{\forall i}\{ITi\}$.

Set $NX_i = i$ dengan nilai IT_{maks}. Jika terdapat lebih dari 1 stasiun kerja yang memiliki nilai IT_{maks}yang sama, maka pilihlah stasiun kerja secara random. Lanjutkan ke langkah 9.

Langkah 9

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

Jika ya, set $VX_j = j$ terpilih dan lanjutkan ke langkah 11.

Jika tidak, lanjutkan ke langkah 10.

Langkah 10

Pilih VX_j, $j \in NXi$ menggunakan persamaan $VXj = arg_{j \in NXi}^{maks} \{t_j\}$ kemudian set j dengan nilai t_j terbesar sebagai VX_j. Jika terdapat nilai $t_j$yang sama, maka pilih secara random dan lanjutkan ke langkah 11.

Langkah 11

Pilih VY_j, $j \notin NX_i$ lakukan proses *exchange*dan lanjutkan ke langkah 11.1.

11.1. Apakah proses exchange melanggar precedence constrain atau VYj sudah pernah menghasilkan konfigurasi yang sama?
Jika ya, lanjutkan ke langkah 11.2.
Jika tidak, lanjutkan ke langkah 11.3.

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.

11.3. Lakukan proses *exchange* untuk semua *j* yang memungkinkan. Gambar *layout* hasil konfigurasi, hitung nilai *EL*&*SI* dan periksa apakah $EL \geq EL_{Th}$?
Jika tidak, kembali ke langkah 11.2.
Jika ya, lanjut ke langkah 11.4.

11.4. Apakah terdapat jumlah VY_j dalam batas $EL \geq EL_{Th} > 1$?
Jika ya, pilih *j* dengan nilai *EL* terbaik, jika terdapat nilai *EL* yang sama pilih nilai *SI* terkecil. Jika terdapat nilai *SI* yang sama pilih secara random dan lanjutkan ke langkah 11.5.
Jika tidak, lanjut ke langkah 11.5.

11.5. Apakah *EL*=100%?
Jika ya, lanjut ke langkah 20.
Jika tidak, ke langkah 13.

Langkah 12

Apakah $e > e_{maks}$?
Jika ya, lanjut ke langkah 13.
Jika tidak, kembali dan ulangi langkah 7.

Langkah 13

Apakah terdapat *j* pada suatu NX_i yang dapat dilakukan proses *intraexchange* tanpa melanggar *precedence constrain*?
Jika ya, lakukan proses *intraexchange*, gambar *layout* hasil konfigurasi, hitung nilai *EL*&*SI* dan set hasil *intraexchange* sebagai *current solution.* Lanjutkan ke langkah 14.
Jika tidak, set hasil *exchange* sebagai *current solutin* lanjutkan ke langkah 14.

Langkah 14

Set $EL_{terbaik}$ dari seluruh solusi yang muncul dan EL_{Th} dan lanjutkan ke langkah 15.

Langkah 15

Set $i = 1$ sebagai NI_i dan $i_{maks} = m$.

Langkah 16

Pilih $j \notin NI_i$ lakukan proses *insert* dan lanjutkan ke langkah 16.1.

16.1. Apakah terdapat *j* yang dilakukan proses *insert* tanpa melanggar *precedence constrain* terhadap NI_i?
Jika ya, lanjutkan ke langkah 16.2.
Jika tidak, set $i = i + 1$, dan lanjutkan ke langkah 16.3.

16.2. Apakah *j* yang di-*insert* terhadap *NIi* sudah pernah menghasilkan konfigurasi yang sama?
Jika ya, set $i = i + 1$, dan kembali ke langkah 16.3.
Jika tidak, lanjut ke langkah 16.4.

16.3. Apakah $i = m + 1$?
Jika ya, lanjut ke langkah 18.
Jika tidak, kembali dan ulangi langkah 16.

16.4. Lakukan proses *insert* untuk semua *j* yang mungkin, gambarkan *layout* hasil konfigurasi dan lanjutkan ke langkah 16.5.

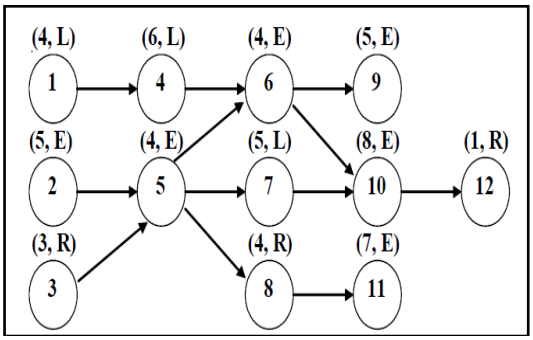
16.5. Apakah jumlah stasiun kerja berkurang setelah dilakukan proses *insert*?
Jika ya, lanjutkan ke langkah 16.8.
Jika tidak, lanjutkan ke langkah 16.6.
16.6. Hitung nilai *EL*&*SI* untuk semua j yang berhasil. Periksa apakah $EL \geq EL_{Th}$?
Jika ya, lanjutkan ke langkah 16.7.
Jika tidak, set $i = i + 1$, kembali ke langkah 16.3.
16.7. Apakah jumlah *j* hasil *insert* > 1?
Jika ya, pilih *j* dengan nilai *EL* terbaik. Jika terdapat nilai *EL* yang sama maka pilihlah nilai *SI* yang terbaik. Jika sama pilih *j* secara random. Lanjutkan ke langkah 16.8.
Jika tidak, lanjutkan ke langkah 16.8.
16.8. Apakah *EL* = 100%?
Jika ya, ke langkah 20.
Jika tidak, ke langkah 17.

Langkah 17
Set konfigurasi hasil *insert* sebagai *current solution* untuk iterasi selanjutnya, set $EL_{terbaik}$ dan EL_{TH} , dan lanjutkan ke langkah 18.
Langlah 18
Set k = k +1.
Langkah 19
Cek apakah k > $\lceil n/2 \rceil$?
Jika ya, lanjutkan ke langkah 20.
Jika tidak kembali ke langkah 6.
Langkah 20
Simpan dan tampilkan konfigurasi terbaik.

4. PENGUJIAN ALGORITMA DAN ANALISIS

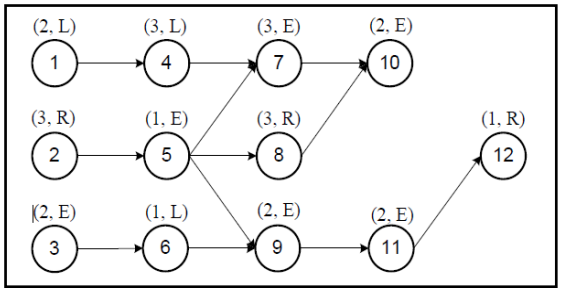
4.1 Pengujian Algoritma

Algoritma V*ariable Neighborhood Decent with Fixed Threshold* (VND *with Fixed Threshold*) diujikan melalui 4 skenario, dan setiap skenario diujikan pada 3 nilai *fixed threshold* yaitu 25%, 15% dan 5%. Pengujian bertujuan melihat apakah algoritma usulan dapat digunakan untuk lintasan perakitan *two-sided*. Skenario 1 adalah set data 12 elemen kerja diambil dari set data milik Kim *et al*. (2000) dengan CT = 22 seperti pada Gambar 1.



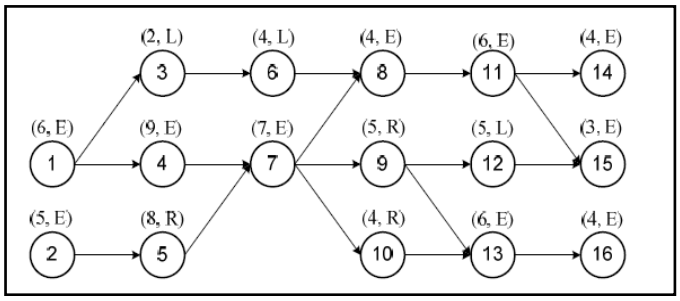
Gambar 1. *Precedence Diagram* Skenario 1 (sumber: Kim *et al,* 2000)

Skenario 2 menggunakan set data 12 elemen kerja dengan CT = 5 yang diambil dari set data Kim *et al*. (2000) seperti pada Gambar 2. Skenario 2 dan skenario 1 memiliki jumlah elemen pekerjaan yang sama, tetapi waktu operasi dan *precedence diagram* yang berbeda.



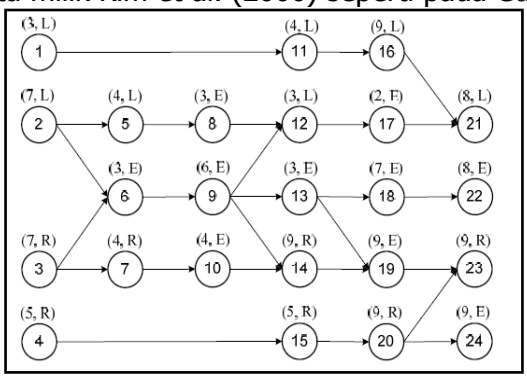
Gambar 2.*Precedence Diagram* Skenario 2
(sumber: Kim *et al,* 2000)

Skenario 3 dengan set data 16 elemen kerja dan CT = 22 diambil dari set data Lee *et al.*(2001) seperti pada Gambar 3.



Gambar 3. *Precedence Diagram* Skenario 3
(sumber : Lee *et al,* 2001)

Skenario 4 mengujikan algoritma usulan pada set data 24 elemen kerja dengan CT = 20 yang diperoleh dari set data milik Kim *et al*. (2000) seperti pada Gambar 4.



Gambar 4. *Precedence Diagram* Skenario 4
(sumber : Kim *et al,* 2000)

Hasil yang diperoleh dari penelitian dapat dilihat pada Tabel 2 dan 3.

Tabel 2. Hasil Pengujian Algoritma

	Hasil Pengujian Algoritma					
	25%		15%		5%	
	EL	*SI*	*EL*	*SI*	*EL*	*SI*
Skenario 1	87,50%	4,359	87,50%	4,359	86,154%	5,1962
Skenario 2	83,333%	3	83,333%	3	83,333%	3
Skenario 3	93,18%	3,742	93,18%	3,742	93,18%	3,742
Skenario 4	87,50%	7,7	87,50%	7,7	87,50%	7,7

Tabel 3. Hasil Perbandingan Algoritma Usulan

Set Data		Jumlah Elemen Kerja (*n*)	*CT*	Optimasi	Heuristik		Metaheuristik									
Heuristik	Meta-Heuristik			Wu *et al.* (2008)	Lee *et al.* (2001)	Waldemar Grzechca (2008)	*Genetic Algorithm*	GRASP			VND			VND *with Fixed* Th		
							Kim *et al.* (2000)	Pratama (2010)			Apnena (2011)			Penelitian		
				Σ Stasiun Kerja (SK)				ΣSK	*EL*	*SI*	ΣSK	*EL*	*SI*	ΣSK	*EL*	*SI*
	Kim *et al.* (2000)	12	16			4 (84,38%)								4	87,50%	4,4
	Kim *et al.* (2000)	12	5	6			6	6	83%	2,6	6	83,33%	2,2	6	83,333%	2,2
Lee *et al. (2001)*		16	22	4	4			4	93,18%	3,7	4	93,18%	3,7	4	93,18%	3,7
	Kim *et al.* (2000)	24	20				8	8	87,50%	9,4	8	87,50%	7,7	8	87,50%	7,7

4.2 Analisis

Berdasarkan Tabel 2, algoritma usulan diujikan pada 4 skenario data yang masing-masing diujikan pada 3 nilai *fixed Threshold*. Perbedaan nilai *fixed Threshold* tidak terlihat pada skenario 2, skenario 3 dan skenario 4. Namun pada skenario 1 terdapat perbedaan hasil. Nilai *fixed threshold* 5% menghasilkan efisiensi lebih kecil dibanding 15% dan 25%. Nilai *fixed threshold* berpengaruh pada ambang batas penerimaan solusi. Semakin besar nilai *fixed threshold* semakin besar nilai ambang batas yang diijinkan untuk menerima solusi baru namun semakin panjang iterasi yang harus diselesaikan. Semakin kecil nilai *fixed threshold* semakin kecil penerimaan solusi baru dan semakin sedikit pula jumlah iterasi.

Algoritma usulan belum pernah diusulkan sebelumnya, sehingga hasil penelitian ini dapat dilakukan perbandingan. Penyelesaian masalah algoritma VND *with Fixed Threshold* melalui perubahan struktur *neighborhood exchange* dan *insert* sama dengan penelitian sebelumnya, namun terdapat perbedaan pada penerimaan solusi karena algoritma usulan mempertimbangkan solusi yang muncul meski bukan solusi yang lebih baik. Perbandingan hasil algoritma usulan dengan penelitian-penelitian sebelumnya ditampilkan pada Tabel 3. Pada skenario 1 jumlah stasiun kerja algoritma usulan dan hasil penelitian Grzechca (2008) memiliki jumlah yang sama, namu nilai efisiensi algoritma usulan lebih baik yaitu 87,50% sedangkan penelitian Grzechca (2008) 84,38%. Skenario 2, skenario 3 dan skenario 4 menghasilkan jumlah stasiun kerja dan nilai efisiensi yang sama dengan penelitian-penelitian sebelumnya. Algoritma usulan untuk skenario 2 dan skenario 4 menghasilkan *smoothnes index* lebih baik dari penelitian Pratama (2010) namun sama dengan hasil penelitian Apnena (2011). Tujuan penelitian ini adalah minimisasi jumlah stasiun kerja untuk mendapatkan nilai efisinsi yang maksimum.

5. KESIMPULAN

Kesimpulan yang diperoleh dari hasil penelitian adalah:

1. Pengembangan algoritma yang diusulkan adalah algoritma *Variable Neighborhood Descent with Fixed Threshold* (VND *with Fixed Threshold*) yang diujikan untuk menyelesaikan masalah pada lintasan perakitan *two-sided* dengan kriteria minimisasi jumlah stasiun kerja.
2. Algoritma usulan memungkinkan ruang penerimaan solusi yang lebih besar dibandingkan penelitian-penelitian sebelum karena mempertimbangkan semua solusi yang muncul meskipun bukan solusi yang lebih baik selama solusi tersebut berada dalam ambang batas penerimaan.
3. Nilai *fixed threshold* berpengaruh pada proses pencarian global optimum. Semakin besar nilai *fixed threshold* semakin besar ruang penerimaan solusi, semakin kecil nilai *fixed threshold* semakin sempit ruang penerimaan solusi.

4. Hasil pengujian algoritma usulan skenario 1, skenario 2, skenario 3 dan skenario 4 menunjukkan jumlah stasiun kerja sama dengan penelitian sebelumnya. Namun pada skenario 1 nilai efisiensi lintasan lebih baik.

DAFTAR PUSTAKA

Apnena, D. R. (2011). *Model Keseimbangan Lintasan Perakitan Two-Sided Menggunakan Algoritma Variable Neighborhood Descent dengan Kriteria Minimisasi Jumlah Stasiun Kerja*. Institut Teknologi Nasional.

Bartholdi, J. J. (1993). Balancing two-sided assembly lines: a case study. *International Journal Prodction Research* 31(10):2447–2461.

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

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

Kim, Y. K., Kim, Y., dan Kim, Y.J. (2000). Two-sided assembly line balancing: a genetic algorithm approach. *Production Planning and Control* 11 (1),44–53.

Lee, T.O., Kim, Y., dan Kim, Y.K. (2001(. Two-sided assembly line balancing to maximize work relatedness and slackness. *Comput Ind Eng* 40 (3),273–292.