© LPPM Itenas | No.1 | Vol. XVI
Januari 2012

Model Keseimbangan Lintas Perakitan Menggunakan Algoritma *Variable Neighborhood Descent* dengan Kriteria Minimasi Stasiun Kerja

ARIF IMRAN, MARTINO LUIS, DANANG RAHADIAN, EMSOSFI ZAINI

Jurusan Teknik Industri, Institut Teknologi Nasional (Itenas) - Bandung
Email: arifimr@yahoo.com

ABSTRAK

Penelitian ini membahas masalah keseimbangan lintasan perakitan sederhana tipe I (Single Assembly Line Balancing Problem I (SALBP I)) menggunakan algoritma Variable Neigborhood Descent (VND) dengan kriteria minimisasi jumlah stasiun kerja. Algoritma VND terdiri dari dua tahap, yaitu tahap pembangkitan solusi inisial dan tahap local search. Solusi awal diperoleh dengan mengaplikasikan algoritma region approach yang kemudian diperbaiki dengan menggunakan neighborhood/local-search seperti 1-0 insertion dan swap (1-1 interchange). Algoritma usulan diuji dengan menggunakan beberapa data set yang terdapat dalam literatur. Hasil pengujian menunjukkan bahwa algoritma usulan dapat menghasilkan solusi yang sama dengan solusi terbaik yang telah dipulikasikan.

***Kata kunci**: assembly line balancing, local search, variable neighborhood, stasiun kerja.*

ABSTRACT

This paper address the Single Assembly Line Balancing Problem I (SALBP I) using the Variable Neigborhood Descent (VND) with minimizing work station number criterion. The VND algorithm consist of two steps, the generation of the initial solution and the improvement step that using several neighborhoods/local searches. The initial solution is obtained by applying the region approach algorithm and then improved by using two neighborhoods/local searches, the 1-0 insertion and the 1-1 inter-change (swap). The proposed algorithm is tested using data sets from literatures. The result shows that the proposed algorithm produces similar results with the best known solution published.

***Keywords**: assembly line balancing, local search, variable neighborhood, work station.*

1. PENDAHULUAN

Simple assembly line balancing problem (SALBP) merupakan permasalahan keseimbangan lintasan perakitan dengan model lintasan tunggal yang memproduksi satu jenis produk yang identik. SALBP dapat dibagi menjadi dua yaitu SALBP I dan SALBP II. SALBP I merupakan permasalahan penugasan sejumlah elemen kerja pada beberapa stasiun kerja untuk minimisasi jumlah stasiun kerja dengan waktu siklus yang telah diketahui. SALBP II merupakan pengalokasian sejumlah elemen kerja pada beberapa stasiun kerja dengan tujuan meminimisasi waktu siklus dengan jumlah stasiun kerja yang telah diketahui. Kedua masalah ini termasuk dalam masalah optimisasi kombinatorial atau dikenal sebagai permasalahan non-deterministik *polynomial hard* (NP-*hard*) [1]. Artinya permasalahan SALBP memerlukan waktu komputasi yang lama untuk mendapatkan solusi yang optimal jika permasalahan bertambah kompleks, misalnya jumlah elemen kerja atau jumlah stasiun kerja bertambah besar.

Terdapat beberapa literatur yang membahas SALBP I di antaranya Johnson yang mengusulkan algoritma FABLE yang berbasis teknik *branch and bound* (B&B) [2]. Hoffmann mengajukan algoritma enumerasi EUREKA [3]. Scholl dan Klein menggunakan prosedur yang berbasis *bi-directional Branch & Bound* [4]. Liu dkk. mengembangkan B&B yang berbasis algoritma *constructive* dan *destructive* dalam menyelesaikan SALBP I [5]. Penelitian-penelitian ini menghasilkan solusi optimal, tetapi tidak efisien dalam penggunaan waktu komputasi. Hal ini ditunjukkan oleh Elsayed dan Boucher dalam menyelesaikan SALBP dengan jumlah elemen kerja, $n \leq 9$ [6]. Oleh karena itu, pendekatan-pendekatan berbasis heuristik dikembangkan dalam menyelesaikan SALBP.

Beberapa penelitian yang menggunakan metode heuristik untuk menyelesaikan SALBP I di antaranya Baybars yang mengembangkan metode yang mendekomposisi permasalahan menjadi beberapa submasalah dan mengkombinasikan beberapa aturan prioritas untuk menyelesaikan submasalah tersebut [7]. Talbot dkk. mengembangkan tiga prosedur heuristik berbasis aturan prioritas *single pass*, komposit heuristik dan aturan *backtracking* [8]. Hackman dkk. mengajukan algoritma B&B yang dikombinasikan dengan beberapa aturan *fathoming* heuristik dalam menyelesaikan SALBP I [9]. Boctor mengusulkan metode heuristik berbasis aturan prioritas *multipass* dan COMSOAL [10]. Ponnambalam dkk. melakukan studi perbandingan dari enam pendekatan heuristik, yaitu *rank positional weight*, Kilbridge dan Wester, Moodie dan Young, Hoffmann *precedence matrix*, *immediate update first fit*, dan *rank and assign* heuristik [11]. Scholl dan Voβ mengusulkan algoritma *tabu search* [12]. Suresh dkk. membahas SALBP I dengan menggunakan algoritma genetik dengan mengizinkan solusi tidak fisibel dan memperbaikinya dengan melakukan proses *exchange* [13]. Sabuncouglu dkk. mengusulkan algoritma genetik dengan melakukan pengembangan pada struktur kromosom dengan teknik partisi pada proses evokasi [14]. Goncalves dan De Almeida mengembangkan algortima genetik campuran [15]. Bautista dan Pereira menggunakan *ant colony system* (ACS) [16]. Lapierre dkk. memodifikasi algoritma *tabu search* dengan mengusulkan intensifikasi dan diversifikasi dari dua *neighbourhood* dan mengizinkan algoritma mengeksplorasi solusi-solusi tidak fisibel [17]. Andres dkk. menerapkan algoritma *greedy randomized adaptive search procedure* (GRASP) [18].

2. METODOLOGI

VND metaheuristik dikembangkan oleh Hansen dan Mladenovic untuk menyelesaikan masalah optimisasi kombinatoris [19]. Ide dasar dari VND adalah perubahan *neighborhood* yang sistematis dalam metode pencarian lokal (*local search*). Algoritma VND usulan dilengkapi dengan beberapa *local search.* Algoritma dasar VND dapat dilihat pada Gambar 1.

Initialization. Select a set of neighbourhood structures N_l, for $k = 1,...,l_{\max}$ that will be used in the search; find an initial solution x;
Repeat the following sequence until no improvement is obtained:
(1) Set $l \leftarrow 1$
(2) *Repeat* the following steps until $l = l_{\max}$:
(a) Exploration of neighbourhood. Find the best neighbour x' of x $(x' \in N_l(x))$;
(b) Move or not. If the solution x' obtained is better than x, set $(x \leftarrow x')$, and $l \leftarrow 1$; otherwise, set $l \leftarrow l + 1$.

Gambar 1. Algoritma VND

Dari studi literatur yang dilakukan, algoritma VND belum pernah digunakan untuk menyelesaikan SALBP I. Oleh karena itu penelitian ini menggunakan algoritma VND untuk menyelesailkan SALBP I dengan kriteria minimisasi jumlah stasiun kerja.

3. MODEL USULAN

Ide dasar dari pengembangan model pada penelitian ini adalah menempatkan sejumlah elemen kerja (*task*) ke dalam beberapa stasiun kerja (*work station*) tanpa melanggar hubungan preseden (*precedence relations*) dan waktu perakitan di setiap stasiun kerja tidak melebihi waktu siklus lintasan perakitan. Proses ini dilakukan untuk mendapatkan konfigurasi elemen kerja dalam stasiun kerja yang memberikan efisiensi lintasan yang tinggi dan *smoothness index* yang rendah. Algoritma VND terdiri dari dua tahap, yaitu tahap pembangkitan solusi inisial dan tahap *local search*.

3.1 Batasan masalah dan asumsi

Batasan-batasan yang digunakan:
1) Jumlah operator di setiap stasiun kerja tidak diperhitungkan.
2) Peralatan atau mesin yang digunakan untuk merakit produk tidak dibahas.
3) Sistem yang dibahas hanya untuk lintasan perakitan produk tunggal.

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) *Precedence diagram* sudah diketahui.
4) Waktu *set up* setiap elemen kerja diasumsikan termasuk ke dalam waktu operasi setiap elemen kerja.

Notasi-notasi yang digunakan adalah sebagai berikut:
CT = Waktu Siklus
SI = *Smoothness Index*
i = Indeks Stasiun Kerja (i = 1, 2, 3, …, m)
j = Indeks Elemen Kerja (j = 1, 2, 3, …, n)
k = Jumlah Stasiun Kerja
t_j = Waktu Proses untuk Elemen Kerja ke-j
Q_i = Akumulasi waktu elemen kerja pada stasiun kerja ke-i
IT = *Idle Time*
I_i = *Idle Time* pada stasiun kerja ke-i
IM = *Idle Time* dengan nilai terbesar
NY_i = *Neighborhood Structure* penukar
NX_i = *Neighborhood Structure* yang ditukar

NI_i = *Neighborhood Structure* pada proses *Insert*
e = Indeks iterasi pada *exchange* (e = 1, 2, 3, …, e_{maks})
r = Indeks iterasi pada *insert* (r = 1, 2, 3, …, r_{maks})
VY_j = Elemen kerja penukar
VX_j = Elemen kerja ditukar

3.2 Algoritma Model Usulan

Langkah-langkah algoritma:

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)

Tahap 2 *Local Search*

Langkah 2

Input data j, t_j, *CT*, *E, SI* dari solusi inisial

Langkah 3

Set $e = 1$

$e_{maks} = m$

Langkah 4

Tentukan *Idle Time* (*IT*) setiap stasiun kerja dengan persamaan

$I_i = {}^{maks}_{\forall_i}\{CT - Q_i\}$

Jika semua I_i telah terpilih maka lanjutkan ke langkah 10

Lainnya, lanjutkan ke langkah 5

Langkah 5

Pilih nilai *IT* yang terbesar (*IM*) dengan menggunakan persamaan $IM = {}^{maks}_{\forall_i}\{I_i\}$ dan set sebagai NX_i. Jika terdapat lebih dari 1 stasiun kerja memiliki nilai *IM* yang sama, pilih stasiun kerja secara random.

Langkah 6

Periksa apakah $|NX_i| = 1$?

Jika **Ya,** set elemen kerja terpilih sebagai VX_j & lanjutkan ke langkah 8

Jika **Tidak**, lanjutkan ke langkah 7

Langkah 7

Pilih $VX_j \in NX_i$ menggunakan persamaan $VX_j = arg\,{}^{maks}_{j \in NX_i}\{t_j\}$

kemudian set sebagai VX_j

Jika terdapat t_j yang sama maka pilih secara random

Periksa apakah masih terdapat elemen kerja $\in NX_i$ yang masih dapat ditukar?

Jika **Ya,** ulangi langkah 7

Jika **Tidak,** maka set $e = e + 1$ dan kembali ke langkah 4

Langkah 8

Pilih *neighbourhood* penukar (NY_i) untuk proses *exchange*. jika stasiun kerja sekitar lebih dari 1 maka tentukan secara random

Langkah 9

Pilih elemen kerja j dari NY_i secara random. Tukarkan elemen kerja dan periksa apakah proses *exchange* melanggar *precedence constraint* atau melebihi waktu siklus?

Jika **Ya,** kembali ke langkah 7

Jika **Tidak,** set $e = e + 1$ dan kembali ke langkah 4

Langkah 10

Apakah $e > e_{maks}$?

Jika **Ya,** lanjutkan ke langkah 11

Jika **Tidak,** kembali dan ulangi langkah 7

Langkah 11

11.1 Set r =1 $r_{maks} = m$

11.2 set $i = r$ dan pilih $j \in NS_{i+1}$ secara random

Lakukan proses *insert* $j \in NI_{i+1}$ terhadap NI_i tanpa melanggar *precedence constraint* dan waktu siklus

Periksa apakah $j \in NI_{i+1}$ sudah melalui proses *insert* seluruhnya?

Jika **Ya,** lanjutkan ke langkah 11.5

Jika **Tidak,** lanjutkan ke langkah 11.3

11.3 Periksa apakah E = 100% ?

Jika **Ya**, simpan konfigurasi penempatan elemen kerja pada setiap stasiun kerja dan lanjutkan ke langkah 12

Jika **Tidak,** lanjutkan ke langkah 11.4

11.4 Periksa apakah nilai *SI* lebih kecil dari *SI current solution*?

Jika **Ya,** simpan konfigurasi penempatan elemen kerja pada setiap stasiun kerja dan perbaharui *SI current solution*, kemudian lanjutkan ke langkah 11.5

Jika **Tidak,** lanjutkan ke langkah 12

11.5 Periksa apakah $r > r_{maks}$?

Jika **Ya,** simpan hasil dan lanjutkan ke langkah 12

Jika **Tidak,** set $r = r + 1$ dan kembali ke langkah 11.2

Langkah 12

Stop dan tampilkan konfigurasi penugasan elemen kerja pada stasiun kerja

Keterangan untuk langkah-langkah dalam algoritma:

1) Tahap 1 *Initial Solution*

Pada tahap ini dilakukan pembentukan sebuah solusi inisial yang menggunakan algoritma *Region Approach* (RA, Kilbridge-Wester). 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 (*E*), dan *smoothness index* (*SI*) dengan menggunakan persamaan sebagai berikut:

Idle time (*IT*)

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

Efisiensi lintasan (*E*)

$$E = \frac{\sum_{i=1}^{m} SK_i}{m \times CT} \quad (2)$$

Smoothness index (*SI*)

$$SI = \sqrt{\sum_{i=1}^{m} (\text{maks}\, SK_i - SK_i)^2} \quad (3)$$

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

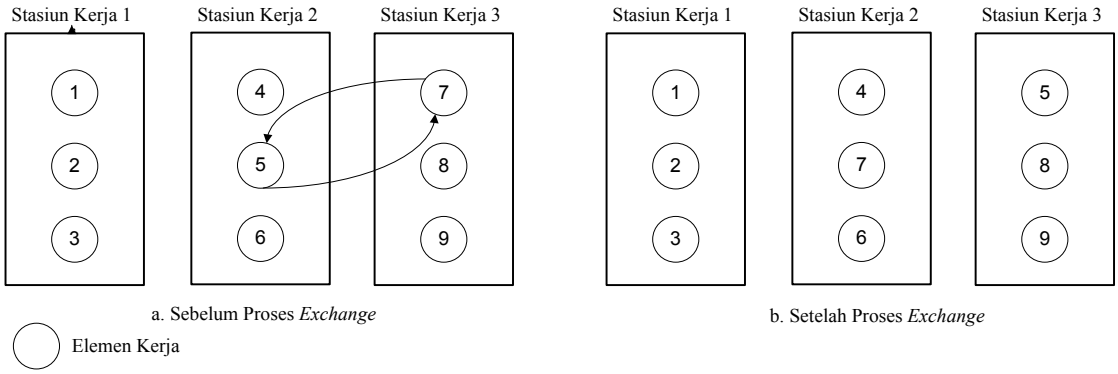
2) Tahap 2 *Local Search*

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:

a. *Exchange*

Teknik *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.

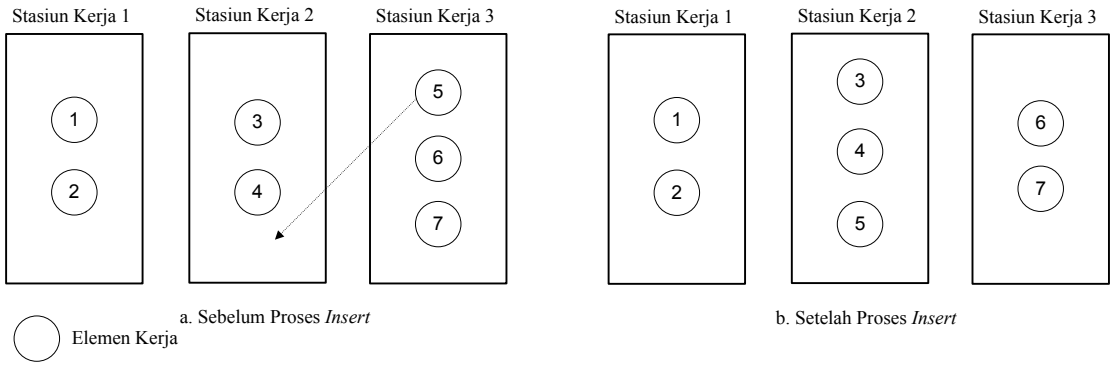


Gambar 2. Proses *Exchange* pada Perubahan Suatu Struktur *Neighbourhood*
a) Sebelum Proses *Insert*
b) Setelah Proses *Insert*

b. *Insert*

Teknik *Insert* merupakan perubahan struktur dengan cara menyisipkan sebuah operasi ke dalam 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 ketujuh 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 3. Proses *Insert* pada Perubahan suatu Struktur *Neighbourhood*
a) Sebelum Proses *Insert*
b) Setelah Proses *Insert*.

4. PENGUJIAN MODEL DAN ANALISIS

Algoritma usulan akan diuji menggunakan data dari Mitchell [20], dengan jumlah elemen kerja, n = 21, Jackson [21], dengan jumlah elemen kerja, n = 11, Rosenberg dan Ziegler [22] dengan jumlah elemen kerja, n = 25, dan data Buxey [23] dengan jumlah elemen kerja, n = 29.

Tabel 1. Perbandingan Hasil Perhitungan Algoritma Usulan dan Beberapa Literatur

Set Data	Jumlah Stasiun Kerja					
	Jumlah Elemen Kerja (n)	Waktu Siklus (CT)	Hoffman (1992)	Scholl (1997)	Goncalves dan De Almeida (2002)	Algoritma Usulan
Jackson (1956)	11	14	4	-	4	4
Mitchel (1957)	21	26	5	-	5	5
Buxery (1974)	29	36	-	10	-	10
Rosenberg (1992)	25	25	-	6	-	6

Dari Tabel 1, dapat dilihat bahwa algoritma usulan menghasilkan solusi yang sama dengan solusi yang tedapat pada literatur. Pada data Mitchell [20] algoritma usulan menghasilkan 5 stasiun kerja. Hasil sama dengan hasil dari Hoffman [3] serta Goncalves dan De Almeida [15]. *CT* awal yang digunakan yaitu 26, sedangkan pada kenyataannya adalah 25. *CT* awal pada algoritma usulan sebesar 26 digunakan karena dapat dilakukan perbandingan hasil penelitian yang telah dipublikasikan lainnya. Sedangkan untuk *CT* awal sebesar 25 tidak bisa dilakukan perbandingan. Data Jackson [21] dengan solusi 4 stasiun kerja dapat dihasilkan dengan menggunakan model usulan VND, hasil tersebut juga sama dengan penelitian Hoffman [3] serta Goncalves dan De Almeida [15]. Adapun data Buxey [23] dan Rosenberg [22] dengan solusi 10 dan 6 stasiun kerja dapat dihasilkan dengan menggunakan model usulan VND. Hasil serupa juga diperoleh Scholl [4].

5. KESIMPULAN DAN SARAN

Dari hasil penelitian dapat disimpulkan bahwa pengujian dengan empat data memperlihatkan algoritma usulan dapat menyelesaikan SALBP I dengan hasil yang sama baiknya dengan penelitian yang dipublikasikan sebelumnya. Hal ini menunjukkan algoritma usulan dapat menghasilkan solusi yang kompetitif.

Untuk penelitian selanjutnya dapat dilakukan minimasi waktu siklus dengan jumlah stasiun kerja yang telah diketahui pada SALBP II. Untuk memperbaiki solusi dapat digunakan *local search* tambahan seperti 2-opt dan GENI, serta dapat juga digunakan algoritmaVNS yang dapat memberikan hasil yang lebih baik daripada algoritma VND.

DAFTAR PUSTAKA

[1] Gutjhar, A.L., dan Nemhauser, G.L., 1964. An Algorithm of the Line Balancing Problem. *Management Science*, 11, 309-315.

[2] Johnson, R.V., 1988. Efficient Modular Implementation of Branch and Bound Algorithms. *Decision Science*, 19, 17-38.

[3] Hoffman, T.R., 1993. Eureka: A Hybrid System for Assembly Line Balancing. *Management Science*, 38, 39-47.

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

[5] Liu, S. B., Ng, K. M., dan Ong, H. L., 2008. Branch-and-Bound Algorithms for Simple Assembly Line Balancing Problem, *International Journal of Advanced Manufacturing Technology*, Vol. 36(1/2), 169-177

[6] Elsayed, E.A., dan Boucher, T.O., 1994. *Analysis an Control of Production System*, 2nd Edition, Prentice Hall, New Jersey.

[7] Baybars, I., 1986. An Efficient Heuristic Method for the Simple Assembly Line Balancing Problem. *International Journal of Production Research*, 24, 149-166.

[8] Talbot, F.B., Patterson, J.H., dan Gehrlein, W.V., 1986. A Comparative Evaluation of heuristic Line Balancing Techniques, *Management Science*, 32, 430-454.

[9] Hackman, S.T., Magazine, M.J., dan Wee, T.S., 1989. Fast, Effective Algorithms for Simple Assembly Line Balancing Problems. *Operations Research*, 32, 916-924.

[10] Boctor, F. F., 1995. A Multiple-Rule Heuristic for Assembly Line Balancing. *Journal of Operations Research Society,* 46(1), 62-69.

[11] Ponanbalam, S.G, Aravindan, P., dan Naidu, G.M., 1999. A Comparative Evaluation of Assembly Line Balancing Heuristics. *International Journal of Advanced Manufacturing Technology*, 15, 557-586.

[12] Scholl, A., dan Vob, S., 1996. Simple Assembly Line Balancing – Heuristic Approaches. *Journal of Heuristics*, 2, 217-244.

[13] Suresh, G., Vinod, V.V., dan Sahu, S., 1996. Genetic Algorithm for Assembly Line Balancing. *Production and Planning Control*, 7, 38-46.

[14] Sabuncouglu, I., Erel, E.M., 2000. Assembly Line Balancing Using Genetic Algorithm. *Journal of Intelligent Manufacturing*, 11, 295-310.

[15] Goncalves, J.F., dan De Almeida, J.R., 2002. A hybrid Genetic Algorithm for Assembly Line Balancing. *Journal of Heuristic*, 8, 629-642.

[16] Bautista, J., dan Pereira, J., 2002. Ant Algorithms for Assembly Line balancing. *Lecture Notes in Computer Science*, 2463, 67-75.

[17] Lapierre, S.D., Ruiz, A., dan Soriano, P., 2006. Balancing Assembly Line with Tabu Search. *European Journal of Operational Research*, 168, 826-837.

[18] Andres, C., Miralles, C., dan Pastor, R., 2008. Balancing and scheduling Task in Assembly Line Balancing, *European Journal of Operational Research*, 187, 1212-1223.

[19] Hansen, P., dan Mladenovic, N., 1999. An Introduction to Variable Neighborhood Search. In Voss,S., Martello, S.,Osman, H., dan Roucairol, C., editor, *Metaheuristics*, Kluwer Academic Publisher.

[20] Mitchell, J., 1957. *A Computational Procedure for Balancing Zoned Assembly Lines*. Research Report 6-94801-1-1-R3, Westing House Research Laboratories, Pittsburgh.

[21] Jackson, J. R., 1956. A Computing Procedure for a Line Balancing Problem. *Management Science*, 2, 261-272.
[22] Rosenberg, O. dan Zielger, H., 1992. A Comparison of heuristics Algorithms for Cost Oriented Assembly Line Balancing. *Zeit-chrift fur Operations Research*, 36, 477-495.
[23] Buxey, G.M., 1974. Assembly Line Balancing with Multiple Stations. *Management Science*, 20,1010-1021.