# PROGRAM KOMPUTASI *RANKED POSITIONAL WEIGHT* UNTUK KESEIMBANGAN LINTASAN PERAKITAN

**Engelina C. Dengah**
Program Studi Teknik Industri
Fakultas Teknik Unika De La Salle Manado
Kampus Kairagi I Kombos Manado
edengah@unikadelasalle.ac.id

## Abstrak

Keseimbangan lintasan perakitan atau *assembly line balancing problem* (ALBP) merupakan masalah yang sering terjadi di teknik industri sebagai bagian dari *NP-hard combinatorial optimization problem*. Dimana tidak terdapat jaminan solusi optimal dalam menyelesaikan permasalahan. ALBP digunakan untuk menyelesaikan masalah optimasi keseimbangan dengan memakai serangkaian elemen kerja yang ditugaskan ke dalam stasiun tertentu berdasarkan berbagai batasan yang sudah diatur. Dibutuhkan masukan untuk mendapatkan hasil yang optimal antara lain elemen kerja, waktu siklus, waktu proses, dan jumlah *predecessor* dari tiap elemen. Data-data tersebut dibutuhkan dengan jumlah yang tidak sedikit dalam implementasi di dunia nyata, sehingga tidak disarankan melakukan perhitungan secara manual karena akan memakan waktu yang lama. Paper ini mengimplementasikan salah satu algoritma lintasan perakitan yaitu *Ranked Positional Weight* (RPW) yang dirancang dengan menggunakan bahasa pemrograman Java. Program ini dapat membaca 50 elemen kerja dengan maksimum 3 *predecessor* untuk setiap elemen yang dieksekusi dalam 3 waktu siklus yang berbeda. Hasil yang didapat berupa perbandingan jumlah stasiun kerja, penugasan elemen kerja di tiap stasiun, dan efisiensi keseimbangan lintasan untuk 3 waktu siklus yang berbeda.

**Kata kunci : *NP-hard combinatorial optimization problem, ranked positional weight*, efisiensi keseimbangan lintasan**

## *Abstract*

*Assembly line balancing (ALBP) is a problem that often occurs in industrial engineering as part of the NP-hard combinatorial optimization problem. Which means no one can guarantee the optimal solution in solving the problems. ALBP used to solve balance optimization problem with a series of elements that are assigned to work in a particular station by the restrictions that have been set. Input needed to obtain optimal results include work element, cycle time, process time for each work element and number of predecessors. These data are needed in large number for the implementation in the real world and is not recommended to do calculations manually because it required long time. This paper implements one of line balancing algorithm i.e. Ranked Positional Weight (RPW) designed using Java programming language. The program can read a maximum of 50 work elements and a maximum of 3 predecessors for each element executed with 3 different times. The results will be the optimum number of work stations, work elements assigned to each station and line balance efficiency for each different cycle time.*

***Keywords : NP-hard combinatorial optimization problem, ranked positional weight, line balance efficiency***

## PENDAHULUAN

Keseimbangan lintasan perakitan berawal dari adanya kombinasi penugasan kerja kepada beberapa operator yang menempati tempat kerja tertentu. Dengan penugasan elemen kerja yang berbeda akan menyebabkan perbedaan dalam jumlah waktu yang tidak produktif. Apabila terjadi hambatan atau ketidakefisienan pada salah satu operator, maka akan mengakibatkan tidak lancarnya aliran material ke operator berikutnya sehingga terjadi waktu menunggu (*delay time*) dan penumpukan material. Tujuan utama yang ingin dicapai adalah mendapatkan tingkat efisiensi yang tinggi bagi setiap operator dan memenuhi rencana produksi yang telah ditetapkan sehingga diupayakan untuk memenuhi perbedaan waktu kerja antar operator dan memperkecil waktu tunggu.

Keseimbangan lintasan (*line* balancing) memiliki beberapa algoritma, yaitu *Computer Method for Sequencing Operations for Assembly Lines* (COMSOAL), *task with Least Number of Predecessors* (TLNP), *Largest Candidate* (LC), dan *Ranked Positional Weight* (RPW). Suwannarongsri dan Puandownreong [5] mengoptimalkan keseimbangan lintasan perakitan menggunakan *tabu search* dengan teknik *partial random permutation*. Perpaduan algoritma ini diprogram menggunakan Matlab. Fonseca et.al., [1] memodifikasi dua dari empat algoritma, yaitu Comsoal dan RPW untuk menyelesaikan masalah keseimbangan lintasan dengan variable waktu *fuzzy*. Peneliti lain seperti Xu dan Xiao [6] mengembangkan suatu model khusus ALBP yang mengintegrasikan simulasi *fuzzy* dan *genetic algorithm* (GA) untuk merancang *hybrid intelligent algorithm*.

Opit et.al., [4] membahas tiga dari empat algoritma, yaitu Comsoal, TLNP dan LC untuk menyelesaikan masalah keseimbangan lintasan dengan menggunakan bahasa pemrograman VBA Excel 2007, VB Database 6.0, dan Visual Studio 2008 C++. Paper ini merupakana implementasi program komputasi untuk RPW untuk menyelesaikan permalasahan keseimbangan lintasan perakitan dengan lebih cepat dan akurat. RPW diprogram menggunakan bahasa pemrograman Java dan dirancang agar pengguna dapat melakukan perubahan pada saat memasukkan data dan terdiri dari 50 elemen kerja dengan maksimum 3 *predecessor* untuk tiap elemennya diuji dengan 3 waktu siklus yang berbeda.

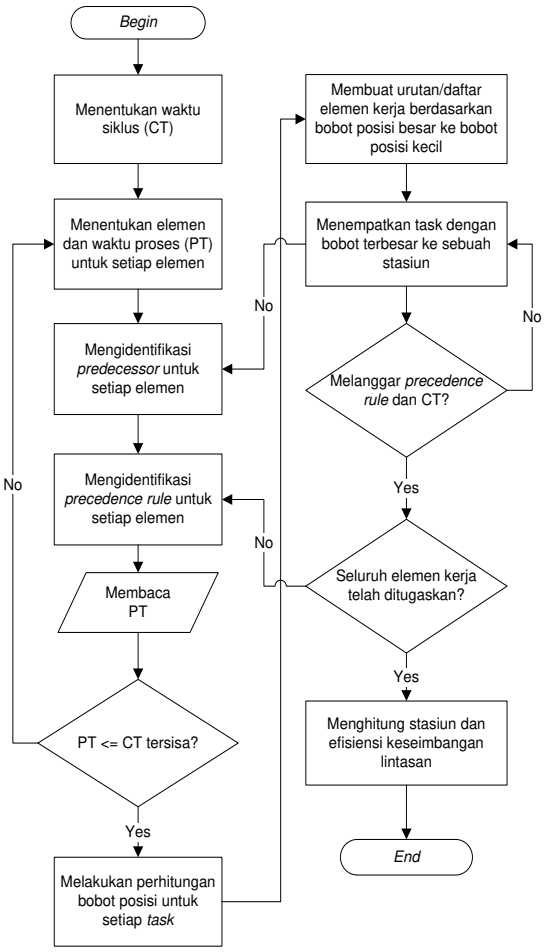
## METODOLOGI

Algoritma RPW menggunakan suatu metode yang digunakan untuk menyeimbangkan lintasan pada proses produksi dengan mengetahui lebih dahulu waktu-waktu yang ada dalam proses perakitan tersebut dengan tujuan agar proses produksi dapat berjalan dengan baik. Malave [3] juga mengungkapkan RPW sebagai metode yang diusulkan oleh Halgeson dan Birnie sebagai pendekatan untuk memecahkan permasalahan pada keseimbangan lintasan dan menemukan solusi dengan cepat.

Urutan langkah-langkah metode RPW menurut Halim [2] adalah sebagai berikut:

1. Lakukan perhitungan bobot posisi untuk setiap *task*. Bobot posisi setiap *task* dihitung dari bobot suatu *task* ditambah dengan bobot *task-task* setelahnya.
2. Lakukan pengurutan *task-task* berdasarkan bobot posisi, yaitu bobot posisi besar ke bobot posisi kecil.
3. Tempatkan *task* dengan bobot terbesar ke sebuah stasiun kerja sepanjang tidak melanggar *precedence constraint* dan waktu stasiun kerja tidak melebihi waktu siklus.
4. Lakukan langkah 3 hingga semua *task* telah ditempatkan kepada suatu stasiun kerja.

Dibawah ini merupakan tahapan atau prosedur dari algoritma RPW.



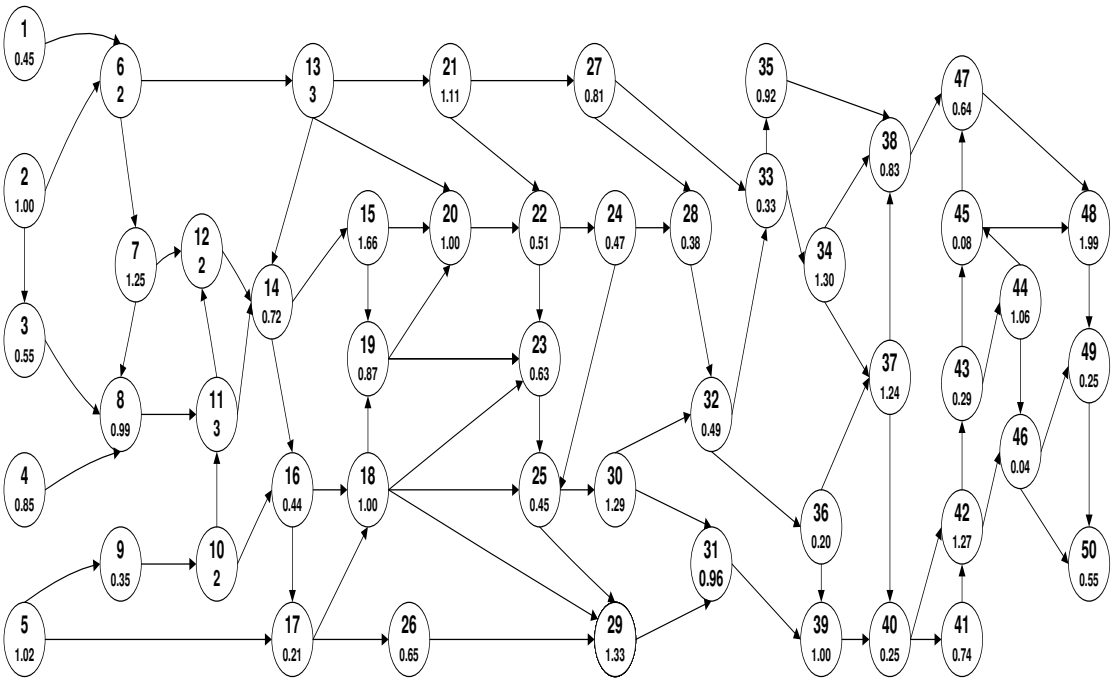

**Gambar 1 Algoritma RPW**

**DATA SET**

Data yang digunakan terdiri atas 50 elemen kerja, waktu proses untuk setiap elemen kerja, dan *predecessor*. Masing-masing elemen kerja memiliki maksimum 3 *predecessor*. Dalam perhitungan program komputasi diberikan beberapa waktu siklus, yaitu 5 menit, 10 menit dan 15 menit. Data yang ada dimasukkan ke dalam program yang telah dirancang dengan menggunakan algoritma RPW. Tabel 1 memuat data dari 50 elemen kerja beserta waktu proses dan *predecessor*. *Precedence diagram* ditunjukkan oleh gambar 2.

**Tabel 1 Data set**

| No. | Elemen Kerja | Waktu (menit) | Predecessor |
|---|---|---|---|
| 1 | 1 | 1 | |
| 2 | 2 | 1 | |
| 3 | 3 | 2 | 2 |
| 4 | 4 | 1 | |
| 5 | 5 | 1 | |
| 6 | 6 | 1 | 2, 1 |
| 7 | 7 | 2 | 6 |
| 8 | 8 | 3 | 7, 4, 3 |
| 9 | 9 | 2 | 5 |
| 10 | 10 | 3 | 9 |
| 11 | 11 | 2 | 10, 8 |
| 12 | 12 | 2 | 11, 7 |
| 13 | 13 | 2 | 6 |
| 14 | 14 | 2 | 13, 12, 11 |
| 15 | 15 | 1 | 14 |
| 16 | 16 | 1 | 14, 10 |
| 17 | 17 | 1 | 16, 5 |
| 18 | 18 | 1 | 17, 16 |
| 19 | 19 | 1 | 18, 15 |
| 20 | 20 | 1 | 19, 15, 13 |
| 21 | 21 | 2 | 13 |
| 22 | 22 | 2 | 21, 20 |
| 23 | 23 | 1 | 19, 18 |
| 24 | 24 | 2 | 22 |
| 25 | 25 | 2 | 24, 23, 18 |
| 26 | 26 | 1 | 17 |
| 27 | 27 | 1 | 21 |
| 28 | 28 | 1 | 27, 24 |
| 29 | 29 | 1 | 26, 25, 18 |
| 30 | 30 | 1 | 25 |
| 31 | 31 | 2 | 30, 29 |
| 32 | 32 | 2 | 30, 28 |
| 33 | 33 | 1 | 32, 27 |
| 34 | 34 | 1 | 33 |
| 35 | 35 | 1 | 33 |
| 36 | 36 | 1 | 32 |
| 37 | 37 | 1 | 36, 34 |
| 38 | 38 | 3 | 37, 35, 34 |
| 39 | 39 | 3 | 36, 31 |
| 40 | 40 | 2 | 39, 37 |
| 41 | 41 | 3 | 40 |
| 42 | 42 | 1 | 41, 40 |
| 43 | 43 | 1 | 42 |
| 44 | 44 | 1 | 43 |
| 45 | 45 | 1 | 44, 43 |
| 46 | 46 | 1 | 44, 42 |
| 47 | 47 | 1 | 45, 38 |
| 48 | 48 | 1 | 47, 45 |
| 49 | 49 | 1 | 48, 46 |
| 50 | 50 | 1 | 49, 46 |

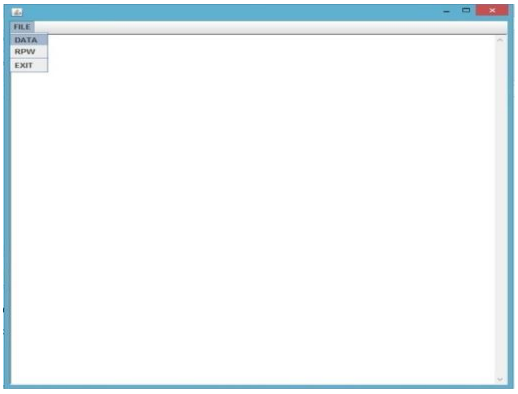



**Gambar 2.** ***Precedence Diagram***

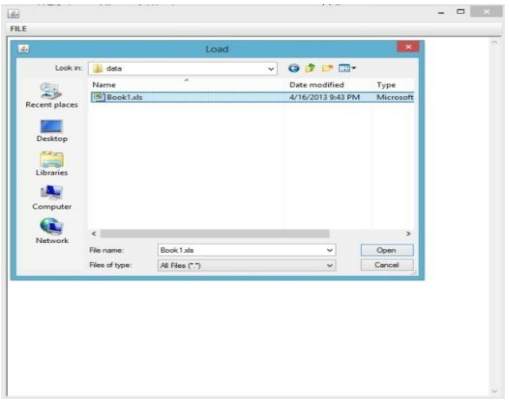
## IMPLEMENTASI

Program komputasi RPW dirancang menggunakan Java dan dijalankan pada PC dengan spesifikasi Intel (R) Core ™ i5-3210M CPU @ 2.50 Ghz dan RAM sebesar 4.00 GB. Dalam pengisian program, dibutuhkan masukkan berupa waktu siklus, elemen kerja, waktu proses, jumlah *predecessor*, dan detail *predecessor* untuk setiap elemen kerja.

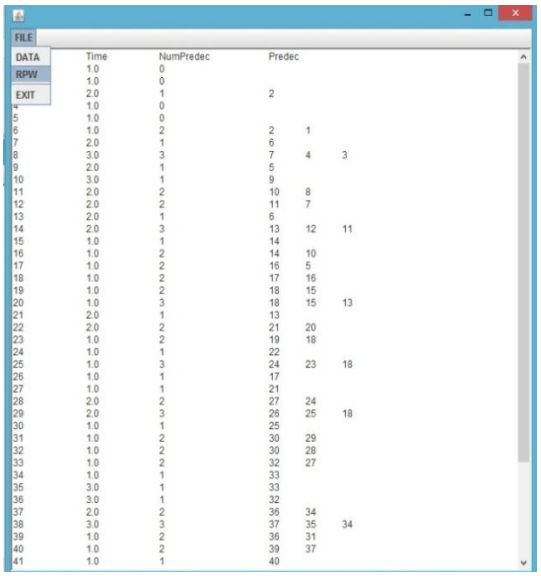
Gambar 3 memperlihatkan tampilan awal program RPW yang memiliki menu *file* berisikan *data*, RPW, dan *exit*. Data yang akan diolah dalam program RPW diambil dengan membuka file excel (lihat gambar 4). Setelah memilih file excel maka secara otomatis setelah file tersebut tercetak di program RPW. Perhitungan RPW dapat dilakukan dengan menekan *sub-menu* RPW pada *menu file* (lihat gambar 5). Tampilan akhir program dapat dlihat pada gambar 6.



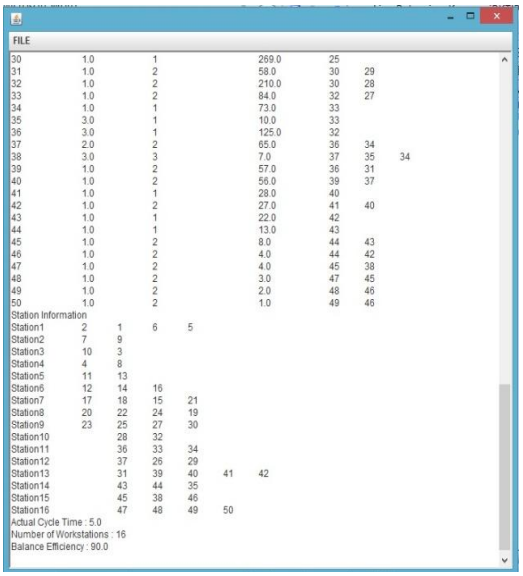

**Gambar 3 Tampilan awal Program RPW**




**Gambar 4 Tampilan pengambilan data**




**Gambar 5 Tampilan perhitungan RPW**




**Gambar 6 Tampilan hasil akhir RPW**

## COMPUTATIONAL RESULTS

Perbandingan hasil akhir algoritma RPW dengan 3 waktu siklus yang berbeda dapat dilihat di tabel 2. Hasil perhitungan RPW untuk waktu siklus ke-3 memiliki persentase efisiensi tertinggi, yaitu 96%. Tidak tertutup kemungkinan hasil persentase yang lebih tinggi dapat diperoleh apabila diberikan waktu siklus yang lebih besar. Sebaliknya jika diberikan waktu siklus yang kecil maka akan diperoleh persentase efisiensi yang rendah. Waktu eksekusi program adalah kurang dari 1 menit untuk masing-masing waktu siklus.

**Tabel 2 Hasil perbandingan Program RPW**

| Waktu Siklus | *Run Time* | Jumlah Stasiun Kerja Optimal | Penugasan Elemen Kerja | Efisiensi |
|---|---|---|---|---|
| 5 menit | < 1 menit | 16 | Stasiun1 = 2, 1, 6, 5<br>Stasiun2 = 7, 9<br>Stasiun3 = 10, 3<br>Stasiun4 = 4, 8<br>Stasiun5 = 11, 13<br>Stasiun6 = 12, 14, 16<br>Stasiun7 = 17, 18, 15, 21<br>Stasiun8 = 20, 22, 24, 19<br>Stasiun9 = 23, 25, 27, 30<br>Stasiun10 = 28, 31<br>Stasiun11 = 36, 33, 34<br>Stasiun12 = 37, 26, 29<br>Stasiun13 = 31, 39, 40, 41, 42<br>Stasiun14 = 43, 44, 35<br>Stasiun15 = 45, 38, 46<br>Stasiun16 = 47, 48, 49, 50 | 90,27% |
| 10 menit | < 1 menit | 8 | Stasiun1 = 2, 1, 6, 5, 7, 9<br>Stasiun2 = 10, 3, 4, 8<br>Stasiun3 = 11, 13, 12, 14, 16, 17<br>Stasiun4 = 18, 15, 21, 20, 22, 24, 19, 23<br>Stasiun5 = 25, 27, 30, 28, 32, 36, 33<br>Stasiun6 = 34, 37, 26, 29, 31, 39, 40, 41<br>Stasiun7 = 42, 43, 44, 35, 45, 38<br>Stasiun8 = 45, 47, 48, 49, 50 | 90,55% |

| Waktu Siklus | *Run Time* | Jumlah Stasiun Kerja Optimal | Penugasan Elemen Kerja | Efisiensi |
|---|---|---|---|---|
| 15 menit | < 1 menit | 5 | Stasiun1 = 2, 1, 6, 5, 7, 9, 10, 3, 4<br>Stasiun2 = 8, 11, 13, 12, 14, 16, 17, 18, 15<br>Stasiun3 = 21, 20, 22, 24, 19, 23, 25, 27, 30, 28, 32<br>Stasiun4 = 36, 33, 34, 37, 26, 29, 31, 39, 40, 41, 42<br>Stasiun5 = 43, 44, 35, 45, 38, 46, 47, 48, 49, 50 | 96% |

## KESIMPULAN

Paper ini mengimplementasikan algoritma RPW ke dalam program komputasi yang dapat digunakan untuk membantu permasalahan ALBP dalam dunia nyata. Menggunakan bahasa pemrograman Java, program ini dirancang sampai dengan 50 elemen kerja dengan maksimum 3 *predecessor* untuk setiap elemen dengan memakai 3 waktu siklus yang berbeda. Hasil akhir yang didapat pengguna diperoleh dalam waktu yang singkat melalui program komputasi tersebut.

Keterbatasan program komputasi ini, antara lain terbatasnya *user interface* program dan tidak tersedia *interface* untuk menampilkan diagram *precedence*. Kedepannya program komputasi ini dapat dikembangkan dengan menampilkan *interface* yang lebih menarik dan dapat menampilkan diagram *precedence*. Selain itu, ke-4 algoritma keseimbangan lintasan dapat disatukan kedalam satu program komputasi. Hal ini memudahkan pengguna untuk dengan cepat membandingkan hasil akhir dari masing-masing algoritma.

## DAFTAR PUSTAKA


1. Fonseca, D.J., C.L. Guest, M. Elam, dan C.L. Karr. 2005. A Fuzzy Logic Approach to Assembly Line Balancing. *Mathware & Soft Computing* 12: 57-74
2. Halim, A.H. 2003. TI-3122 Perencanaan dan Pengendalian Produksi: Keseimbangan Lintasan. Institut Teknologi Bandung.
3. Malave, Cesar. 2000. "Approach to Line Balancing Comsoal & RPW. Texas A&M University.
4. Opit, P.F., M.T. Kornelis, K.A. Mahardini, 2012. Implementasi Program Komputasi Algoritma Keseimbangan Lintasan Perakitan. Konvensi Teknik Industri.
5. Suwannarongsri, S. dan D. Puandownreong. 2008. "Optimal assembly line balancing using tabu search with partial random permutation technique". International Journal of Management Science and Engineering Management Vo. 3 (2008) No. 1, pp. 3-18.
6. Xu, Wieda dan Tianyuan Xiaou. 2008. "Mixed Model Assembly Line Balancing Problem with Fuzzy Operation Times and Drifting Operations." *IEEE Proceedings of the 2008 Winter Simulation Conference*: 1752-1760.