PENGEMBANGAN ALGORITMA *NAWAZ ENSCORE HAM* UNTUK *PERMUTATION FLOWSHOP PROBLEM*

SKRIPSI

Diajukan untuk memenuhi salah satu syarat guna mencapai gelar
Sarjana dalam bidang ilmu Teknik Industri

Disusun oleh :

Nama : Bevinda Juliani Enge

NPM : 2015610162

PROGRAM STUDI SARJANA TEKNIK INDUSTRI

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS KATOLIK PARAHYANGAN

BANDUNG

2019

FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS KATOLIK PARAHYANGAN
BANDUNG

Nama	: Bevinda Juliani Enge
NPM	: 2015610162
Program Studi	: Teknik Industri
Judul Skripsi	: PENGEMBANGAN ALGORITMA *NAWAZ ENSCORE HAM* UNTUK *PERMUTATION FLOWSHOP PROBLEM*

TANDA PERSETUJUAN SKRIPSI

Bandung, Agustus 2019

Ketua Program Studi Sarjana Teknik Industri

(Romy Loice, S.T., M.T.)

Pembimbing Pertama

(Yani Herawati, S.T., M.T.)

Pembimbing Kedua

14/8-19

(Fran Setiawan, S.T., M.Sc.)

Program Studi Teknik Industri
Fakultas Teknologi Industri
Universitas Katolik Parahyangan

Pernyataan Tidak Mencontek atau Melakukan Tindakan Plagiat

Saya, yang bertanda tangan dibawah ini,

Nama : Bevinda Juliani Enge
NPM : 2015610162

dengan ini menyatakan bahwa Skripsi dengan judul :

"PENGEMBANGAN ALGORITMA *NAWAZ ENSCORE HAM* UNTUK *PERMUTATION FLOWSHOP PROBLEM*"

adalah hasil pekerjaan saya dan seluruh ide, pendapat atau materi dari sumber lain telah dikutip dengan cara penulisan referensi yang sesuai.

Pernyataan ini saya buat dengan sebenar-benarnya dan jika pernyataan ini tidak sesuai dengan kenyataan, maka saya bersedia menanggung sanksi yang akan dikenakan kepada saya.

Bandung, 14 Agustus 2018

Bevinda Juliani Enge
2015610162

ABSTRAK

Permutation Flowshop Scheduling Problem (PFSP) adalah masalah penjadwalan yang terjadi pada aliran produksi yang memiliki urutan mesin yang sama untuk setiap pekerjaan. PFSP sendiri termasuk ke dalam *NP-Hard* (*Non-deterministic Polynomial-time Hardness*) sehingga diperlukan metode heuristik untuk melakukan penyelesaiannya. Terdapat beberapa metode heuristik yang dapat digunakan dalam menyelesaikan PFSP dengan ukuran performansi minimasi *makespan.* Metode heuristik untuk meminimasi *makespan* yang dapat digunakan dalam menyelesaikan PFSP adalah algoritma *Palmer, Campbell Dudek Smith* (CDS), *Rapid Access* (RA), *Nawaz Enscore Ham* (NEH), dan *Gupta Chauhan.* Dari metode-metode tersebut, algoritma NEH merupakan algoritma terbaik dalam menyelesaikan PFSP. Sehingga banyak penelitian yang dilakukan untuk meningkatkan performansi dari algoritma NEH.

Pada penelitian ini, akan dilakukan pengembangan algoritma baru untuk menyelesaikan PFSP. Algortima baru yang akan dikembangkan ini berbasiskan algoritma NEH dan mengubah *priority rule* serta *tie-breaking rule*-nya. Hal ini dilakukan untuk meningkatkan performansi yang dapat dihasilkan oleh algoritma NEH. Di penelitian ini, *priority rule* dari algoritma NEH akan diganti menggunakan algoritma *Gupta Chauhan* karena algoritma ini dapat memberikan urutan awal untuk algoritma lain dan *tie-breaking rule*-nya akan diganti menggunakan *tie-breaking rule* dari NEHFF karena algoritma ini memiliki performansi yang paling baik untuk *tie-breaking rule*.

Algoritma baru ini diimplementasikan ke dalam 12 kasus yang terdiri dari 10 *problem instance.* Pada penelitian ini juga dilakukan perbandingan performasi yang dihasilkan oleh algortima baru dengan algoritma pembanding yaitu algoritma *Gupta Chauhan, Palmer,* CDS, RA, NEH, dan NEHFF. Hasil implementasi menunjukan bahwa algoritma baru dapat memberikan performansi yang lebih baik dari algoritma *Gupta Chauhan, Palmer,* CDS, dan RA. Akan tetapi, algortima baru memberikan performansi yang lebih rendah dari NEH dan NEHFF pada kasus ke-9, ke-11, dan ke-12. Sedangkan pada kasus lainnya untuk beberapa *problem instance*, hasil dari algortima baru sama atau lebih baik dari NEH dan NEHFF.

ABSTRACT

Permutation Flowshop Scheduling Problem (PFSP) is a scheduling problem that occurs in a production stream that has the same engine sequence for each job. PFSP itself is included in NP-Hard (Non-deterministic Polynomial-time Hardness) so heuristic methods are needed to solve it. There are several heuristic methods that can be used in solving PFSP using minimization makespan for performance measure. The heuristic method for minimize makespan that can be used in solving the PFSP is Palmer's algorithm, Campbell Dudek Smith (CDS), Rapid Access (RA), Nawaz Enscore Ham (NEH), and Gupta Chauhan. From these methods, the NEH algorithm is the best algorithm in solving the PFSP. So that a lot of research is done to improve the performance of the NEH algorithm.

In this study, a new algorithm will be developed to solve the PFSP. The new algorithm that will be developed is based on the NEH algorithm and changes its priority rule and tie-breaking rule. This is done to improve the performance that can be gived by the NEH algorithm. In this study, the priority rule of the NEH algorithm will be replaced using the Gupta Chauhan algorithm because this algorithm can provide the initial sequence for other algorithms and its tie-breaking rule will be replaced using the tie-breaking rule of NEHFF because this algorithm has the best performance of tie-breaking rule.

This new algorithm is implemented in 12 cases consisting of 10 problem instances. In this study also performed comparisons performance for the new algorithm with comparative algorithms that is Gupta Chauhan algorithm, Palmer, CDS, RA, NEH, and NEHFF. The implementation results show that the new algorithm can provide better performance than the Gupta Chauhan algorithm, Palmer, CDS, and RA. However, the new algorithm gives lower performance than NEH and NEHFF in the 9th, 11th, dan 12th case. Where in other cases for some problem instances, the results of the new algorithm has the same or better result than NEH and NEHFF.

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan karunia yang telah diberikan kepada penulis selama proses penelitian dan juga penyusunan laporan skripsi dengan judul Pengembangan Algoritma *Nawaz Enscore Ham* Untuk *Permutation Flowshop Problem* hingga selesai. Penulis mengucapakan banyak terima kasih kepada orang tua dan keluarga dari penulis yang telah memberi dukungan dan doa selama penyusunan laporan skripsi ini. Dalam proses penyusunan dan penyelesaian laporan skripsi ini, penulis juga mendapat banyak dukungan dan bantuan dari berbagai pihak. Oleh karena itu, tidak lupa penulis mengucapkan banyak terima kasih kepada:

1. Ibu Yani Herawati, S.T., M.T. selaku dosen pembimbing pertama yang telah membimbing, memberikan masukan, dan motivasi selama proses penyusunan laporan skripsi ini.
2. Bapak Fran Setiawan, S.T., M. Sc. selaku dosen pembimbing kedua yang telah membimbing, memberikan masukan, dan motivasi selama proses penyusunan laporan skripsi ini.
3. Ibu Cynthia Prithadevi Juwono, Ir. M.S. dan Ibu Cherish Rikardo, S.Si., M.T. selaku dosen penguji proposal skripsi yang telah memberikan masukan dan perbaikan terhadap proposal skripsi.
4. Bapak Dr. Sugih Sudharma Tjandra, S.T., M.Si. dan Bapak Hanky Fransiscus, S.T., M.T. selaku dosen penguji skripsi yang telah memberikan masukan dan perbaikan terhadap skripsi ini.
5. Oktavianus Roothree dan Tri Lastiur Magdalena Manurung selaku teman baik penulis yang telah mendukung, memotivasi, dan mendoakan penulis selama pengerjaan skripsi ini.
6. Demas, Kevin, dan Junaidi yang telah memberikan bantuan dan masukan dalam proses penyusunan skripsi ini.
7. Dio, Alvin, Ivan, Christoper, Ferdy, dan Rama sebagai teman penulis yang telah memberikan bantuan dan dukungan selama penulisan skripsi ini.
8. Teman-teman Biro Liturgi Kelompok Pelayanan GEMA 2019 yang telah memberi dukungan kepada penulis selama penyusunan skripsi ini.

9. Para pastor dan frater Biara Pratista Kumara Wara Bharata yang telah mendoakan dan memotivasi penulis selama penulisan laporan skripsi ini.
10. Pihak-pihak lainnya yang telah membantu penulis baik secara langsung maupun tidak langsung yang tidak dapat disebutkan satu per satu.

Penulis menyadari masih adanya kelemahan yang terdapat di dalam laporan skripsi ini. Untuk itu penulis dengan senang hati menerima kritik dan saran terhadap laporan skripsi ini sehingga dapat menjadikan laporan skripsi yang lebih baik. Akhir kata, penulis berharap laporan skripsi ini bermanfaat bagi pembaca dan pihak lainnya.

Bandung, 29 Juli 2019

Penulis

DAFTAR ISI

ABSTRAK ..... **i**
ABSTRACT ..... **ii**
KATA PENGANTAR ..... **iii**
DAFTAR ISI ..... **v**
DAFTAR TABEL ..... **vii**
DAFTAR GAMBAR ..... **ix**

BAB I PENDAHULUAN **I-1**
I.1 Latar Belakang Masalah I-1
I.2 Identifikasi dan Rumusan Masalah I-3
I.3 Pembatasan Masalah dan Asumsi Penelitian I-8
I.5 Manfaat Penelitian I-9
I.6 Metodologi Penelitian I-9
I.7 Sistematika Penulisan I-12
BAB II TINJAUAN PUSTAKA **II-1**
II.1 *Scheduling* II-1
II.2 *Production Process Designs* II-3
II.2.1 *Flowshop* II-3
II.2.2 *Jobshop* II-4
II.2.3 *Fixed Site* II-4
II.3 Model Matematis dari *Permutation Flowshop Scheduling Problem* II-5
II.4 Metode Heuristik II-6
II.5 Algoritma *Gupta Chauhan* II-7
II.6 Algoritma *Nawaz Enscore Ham* (NEH) II-8
II.7 Algoritma NEHFF II-9
II.8 Verifikasi dan Validasi II-11
II.9 *Relative Percentage Deviation* dan Average *Relative Percentage Deviation* II-13

BAB III PERANCANGAN DAN IMPLEMENTASI ALGORITMA III-1

III.1 Perancangan Algoritma Baru.....III-1

III.1.1 Notasi Dalam Algoritma Baru III-2

III.1.2 Algoritma Utama III-3

III.1.3 Algoritma Urutan Awal III-8

III.1.4 Algoritma *Makespan* III-14

III.1.5 Algoritma *Tie-breaking*.....III-15

III.1.6 Algoritma NEHFF.....III-17

III.1.7 Algoritma *Idle Time* III-19

III.1.8 Validasi Algoritma Baru.....III-22

II.2 Implementasi Algoritma Baru.....III-57

III.2.1 Verifikasi dan Validasi Program.....III-58

III.2.2 Implementasi Program.....III-65

BAB IV ANALISIS..... IV-1

IV.1 Analisis Perancangan Algoritma..... IV-1

IV.2 Analisis Implementasi Algoritma..... IV-2

IV.3 Analisis Hasil Evaluasi Algoritma Baru dengan Algoritma Pesaing IV-3

BAB V KESIMPULAN DAN SARAN..... V-1

V.1 Kesimpulan V-1

V.2 Saran..... V-2

DAFTAR PUSTAKA

RIWAYAT HIDUP PENULIS

DAFTAR TABEL

Tabel I.1 Perkembangan Algortima NEH I-4
Tabel I.2 *Priority Rule* dan *Tie-breaking Rule* pada Algoritma NEH I-5
Tabel I.3 Perbandingan *Priority Rule* I-6
Tabel I.4 Perbandingan Hasil Algoritma *Gupta Chauhan*, *Palmer*, CDS, dan RA I-7
Tabel I.5 Perbandingan *Tie-breaking Rule* I-7
Tabel III.1 Kasus Sederhana PFSP II-22
Tabel III.2 Hasil *Makespan* dan RPD untuk Setiap *Problem Instance* II-66
Tabel III.3 Hasil ARPD untuk Setiap Kasus II-72

DAFTAR GAMBAR

Gambar I.1 Metodologi Penelitian I-11
Gambar II.1 Macam-Macam *Idle Time* II-10
Gambar II.2 Model Sederhana Simulasi..... II-12
Gambar III.1 Flowchart Algoritma Utama III-7
Gambar III.2 Flowchart Algoritma Urutan Awal III-12
Gambar III.3 Flowchart Algoritma *Makespan* III-15
Gambar III.4 Flowchart Algoritma *Tie-breaking*..... III-16
Gambar III.5 Flowchart Algoritma NEHFF..... III-18
Gambar III.6 Flowchart Algoritma *Idle Time* III-21
Gambar III.7 Program untuk Algoritma Utama III-59
Gambar III.8 Program untuk Algoritma Urutan Awal..... III-60
Gambar III.9 Program untuk Algoritma *Makespan* III-62
Gambar III.10 Program untuk Algoritma *Tie-breaking* III-63
Gambar III.11 Program untuk Algoritma NEHFF III-63
Gambar III.12 Program untuk Algoritma *Idle Time*..... III-64
Gambar III.13 Hasil dari Program..... III-65
Gambar III.14 Perbandingan ARPD dengan Jumlah Pekerjaan..... III-73
Gambar III.15 Perbandingan ARPD dengan Jumlah Mesin III-73

BAB I
PENDAHULUAN

Pada bab ini dijelaskan latar belakang masalah, identifikasi dan perumusan masalah, pembatasan masalah dan asumsi penelitian, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan. Penjelasan tersebut akan dibagi ke dalam beberapa subbab berikut.

I.1 Latar Belakang Masalah

Dalam produksi sebuah produk, perusahaan mengalami masalah penjadwalan. Masalah penjadwalan adalah masalah dalam menentukan urutan (peringkat atau kepentingan) dari suatu kumpulan pekerjaan (tugas atau produk) di mana melalui proses dari satu mesin ke mesin lainnya (Nawaz, Enscore, dan Ham, 1983). Untuk menangani masalah penjadwalan tersebut, perlu dilakukan proses penjadwalan produksi dengan baik oleh perusahaan. Penjadwalan adalah melakukan alokasi sumber daya dari waktu ke waktu untuk menjalankan sekumpulan tugas atau operasi (Baker, 1974). Melakukan penjadwalan dengan baik dapat dilakukan dengan mempertanyakan dua hal, yaitu sumber daya mana yang harus dialokasikan untuk menyelesaikan pekerjaan yang ada dan kapan setiap pekerjaan harus dilakukan (Baker dan Trietsch, 2009). Dengan begitu, penjadwalan yang baik dapat dilakukan dan dapat mencapai tujuan penjadwalan.

Tujuan dari penjadwalan berupa meningkatkan utilisasi sumber daya yang dimiliki dengan mengurangi waktu menganggur pada sumber daya, meningkatkan utilisasi sumber daya dengan mengecilkan waktu yang dibutuhkan untuk menyelesaikan pekerjaan atau minimasi *makespan*, mengurangi *in-process inventory* atau rata-rata pekerjaan yang menunggu untuk dikerjakan dengan mengurangi rata-rata *flowtime*, dan mengurangi *tardiness* yang terjadi dengan mengurangi maximum keterlambatan atau mengurangi pekerjaan yang terlambat (Bedworth dan Bailey, 1987). Untuk mencapai tujuan tersebut, perlu ditentukan ukuran performansi atau *objective function* yang digunakan atau ingin dicapai. Ukuran performansi yang digunakan menentukan tingkat performansi dari jadwal yang dihasilkan dan juga menentukan metode penjadwalan yang digunakan.

Dalam menentukan metode penjadwalan juga, harus dipertimbangkan jenis aliran proses produksi yang digunakan oleh perusahaan.

Aliran proses produksi sendiri ada tiga macam, yaitu *flowshop, jobshop,* dan *fixed site* (Fogarty, Blackstone, dan Hoffmann,1991). Dari ketiga macam aliran tersebut, banyak penelitian yang dilakukan untuk penjadwalan *flowshop.* Hal ini karena penjadwalan *flowshop* merupakan bidang penelitian yang menarik di bidang manufaktur baik ditinjau dari sisi studi ataupun pengaplikasian pada industri atau dunia nyata (Yenisey dan Yagmahan, 2014). Pengaplikasian pada industri dapat ditemukan pada perusahaan otomotif (Liu, Jin, dan Price, 2016) dan perusahaan garmen (Erseven, Akgün, Karakaş, Yarikcan, Yücel, dan Öner, 2018). Aliran *flowshop* pun terbagi menjadi tiga macam, yaitu *permutation flowshop, reentrant flowshop*, dan *hybrid flowshop.* Dari aliran *flowshop* tersebut, *permutation flowshop* banyak dipelajari dan memiliki aplikasi penting dalam sistem manufaktur dan layanan (Erseven et al., 2018).

Permutation flowshop sendiri memiliki ciri khas di mana urutan mesin untuk setiap pekerjaan sama, setiap mesin hanya dapat melakukan satu buah pekerjaan, pekerjaan yang sedang dilaksanakan tidak dapat diinterupsi, dan ketika pekerjaan selesai pada satu mesin maka mesin dapat digunakan untuk mengerjakan pekerjaan selanjutnya (Emmons dan Vairaktarakis, 2013). Pada *permutation flowshop* terdapat masalah penjadwalan yang sering dikenal dengan *permutation flowshop scheduling problem* (PFSP) (Dong, Huang, dan Chen, 2007). Contoh kasus PFSP yang terjadi di dunia nyata seperti penjadwalan untuk pengujian kualitas kain pada perusahaan gramen di mana setiap contoh kain yang ada harus melewati rangkaian pengujian kain yang sama (Erseven et al., 2018). Untuk mengatasi hal tersebut, dibutuhkan metode penjadwalan yang tepat sehingga semua pekerjaan (contoh kain yang harus diperiksa) dapat dijadwalkan pada setiap mesin (pengujian kain yang harus dilakukan) dengan sebaik mungkin hingga didapatkan urutan pekerjaan terbaik.

Untuk mendapatkan urutan pekerjaan terbaik, pada PFSP terdapat ukuran performansi atau *objective function* yang sering digunakan yaitu minimasi *makespan.* Ukuran performansi minimasi *makespan* paling umum digunakan walau terdapat kriteria atau ukuran performansi lain yang dapat digunakan. Ukuran performansi minimasi *makespan* banyak digunakan karena dapat mengurangi waktu penyelesaian dari sebuah produksi (Liu et al., 2016) dan meningkatkan

utilisasi sumber daya yang dimiliki (Bedworth dan Bailey, 1987). Di mana kebanyakan perusahaan menginginkan untuk menggunakan sumber dayanya semaksimal mungkin dan waktu produksi yang diperlukan sesingkat mungkin sehingga dapat menghindari terjadinya keterlambatan. Berdasarkan aliran proses produksi dan ukuran performansi yang digunakan, terdapat beberapa metode penjadwalan yang dapat digunakan.

Akan tetapi, untuk PFSP hanya terdapat beberapa metode yang dapat digunakan. Hal ini dikarenakan PFSP termasuk kedalam *NP-hard (Non-deterministic Polynomial-time Hardness)* (Garey dan Johnson, 1979). *NP-hard* memiliki arti semakin bertambahnya ukuran solusi maka waktu untuk menemukan solusi akan tumbuh secara signifikan (Woeginger, 2001). Penyelesaian *NP-hard* dengan metode eksak akan memakan waktu yang lama sehingga lebih baik menggunakan metode heuristik untuk penyelesaiannya (Paschos, 2009). Oleh karena itu, metode heuristik banyak dikembangkan untuk menyelesaikan permasalahan *NP-hard* (Fernandez-Viagas dan Farminan, 2014). Metode heuristik adalah prosedur penyelesaian masalah atau aturan praktik yang telah terbukti dapat menghasilkan solusi yang baik akan tetapi tidak menjamin solusi tersebut merupakan solusi yang optimal (Bedworth dan Bailey, 1987).

Metode heuristik untuk penjadwal sudah banyak dikembangkan terutama metode heuristik untuk PFSP dengan ukuran performansi minimasi *makespan.* Pengembangan yang dilakukan ini bertujuan untuk meningkatkan performansi yang dapat diberikan oleh metode heuristik tersebut. Miningkatkan performansi yang dapat diberikan oleh metode heuristik penting. Hal ini dikarenakan dengan meningkatnya performansi yang diberikan oleh metode heuristik berarti metode tersebut dapat memberikan urutan pekerjaan atau jadwal pekerjaan yang lebih baik sesuai dengan ukuran performansi yang digunakan dan semakin baik pula dalam menyelesaikan PFSP. Sehingga pengembangan metode heuristik perlu dilakukan supaya hasil atau solusi untuk menyelesaikan PFSP menggunakan metode heuristik semakin baik lagi dan dapat diterapkan pada kasus PFSP di dunia nyata.

I.2 Identifikasi dan Rumusan Masalah

Dari banyaknya pengembangan metode heuristik, metode heuristik untuk *permutation flowshop scheduling problem* (PFSP) dengan ukuran performansi

minimasi *makespan* yang sudah ada adalah algoritma *Palmer, Campbell Dudek Smith* (CDS), *Rapid Access* (RA), *Nawaz Enscore Ham* (NEH), dan *Gupta Chauhan* (Gupta dan Chauhan, 2015). Dari metode heuristik tersebut, algortima NEH merupakan algoritma terbaik dalam menyelesaikan PFSP (Liu et al., 2016). Sudah banyak penelitian yang dilakukan untuk mengembangkan algoritma NEH. Pengembangan yang dilakukan ini bertujuan untuk meningkatkan performansi yang diberikan atau dihasilkan oleh algoritma NEH. Pada Tabel I.1 dapat dilihat perkerbangan dari algortima NEH.

Tabel I.1 Perkembangan Algortima NEH

Algoritma	Tahun	Pengembang
NEH	1983	M. Nawaz, E. Emory Enscore Jr., dan Inyong Ham
NEHNM	2002	M. S. Nagano dan J. V. Moccellin
MNEH	2004	Y. C. Low, K. T. Fang, dan B. Lin
NEHKK	2007	Pawel J. Kalczynski dan Jerzy Kamburowski
NEHKK1	2007	Pawel J. Kalczynski dan Jerzy Kamburowski
NEH-D	2008	Xingye Dong, Houkuan Huang, dan Ping Chen
NEHKK2	2009	Pawel J. Kalczynski dan Jerzy Kamburowski
NEHFF	2014	Victor Fernandez-Viagas dan Jose M. Framinan
NEHLJP	2016	W. Liu, Y. Jin, dan M. Price

Berdasarkan Tabel I.1, dapat dilihat bahwa algoritma NEH masih dikembangkan hingga tahun 2016. Hal ini menunjukan bahwa algoritma NEH merupkan salah satu metode heuristik yang masih menarik untuk dilakukan pengembangan. Selain itu, algoritma NEH merupakan algoritma yang masih digunakan dalam menyelesaikan PFSP. Contohnya penggunaan algoritma NEH untuk menjadwalkan pengujian kain pada perusahaan garmen (Erseven et al., 2018). Karena itu, perlu dilakukan penelitan untuk mengetahui apakah algoritma NEH dapat dikembangkan lagi dengan harapan performansi yang diberikan lebih baik. Pada penelitian ini, akan dilakukan pengembangan algoritma baru yang berbasis pada algoritma NEH.

Algoritma NEH terdiri dari dua bagian yaitu *priority rule* dan *tie-breaking rule. Priority rule* merupakan bagian algoritma yang berfungsi untuk menentukan susunan awal atau *initial sequence* dari pekerjaan yang nantinya digunakan untuk mencari susunan pekerjaan sesuai dengan ukuran performansi yang digunakan atau ingin dicapai. Sedangkan *tie-breaking rule* merupakan bagian algoritma yang berfungsi untuk menentukan urutan pekerjaan mana yang dipilih bila terjadi urutan

pekerjaan dengan hasil yang sama. Pengembangan algoritma NEH biasanya dilakukan pada *priority rule* dan/atau pada *tie-breaking rule*-nya. Pada Tabel I.2 dapat dilihat *priority rule* dan *tie-breaking rule* dari setiap pengembangan algoritma NEH yang sudah ada.

Tabel I.2 *Priority Rule* dan *Tie-breaking Rule* pada Algoritma NEH

Algoritma	***Priority Rule***	***Tie-breaking Rule***
NEH	Total waktu proses disusun dari yang besar ke kecil	Posisi pertama yang muncul dengan hasil sama yang akan dipilih
NEHHM	Perbedaan antara total waktu proses dengan *lower bound* dari waktu mengunggu pekerjaan disusun dari yang besar ke kecil	Posisi pertama yang muncul dengan hasil sama yang akan dipilih
MNEH	Jumlah waktu pemrosesan buatan disusun dari yang besar ke kecil	Posisi dengan waktu menganggur paling kecil di mesin yang *bottleneck* yang akan dipilih
NEHKK	Total waktu proses disusun dari yang besar ke kecil	Pilih posisi dengan waktu penyelesaian maksimum yang paling kecil
NEHKK1	Urutkan dari besar ke kecil berdasarkan jumlah pembobotan waktu pemrosesan (c_j). c_j = min (a_i, b_i), di mana $a_i = \sum_{k=1}^{m}\left[\frac{(m-1)(m-2)}{2}+m-k\right]t_{j,k}$ $b_i = \sum_{k=1}^{m}\left[\frac{(m-1)(m-2)}{2}+k-1\right]t_{j,k}$	Pekerjaan x akan dimasukan di awal jika $a_x \le b_x$. Pekerjaan x akan dimasukan di akhir jika $a_x \ge b_x$.
NEH-D	Jumlah rata-rata dan standar deviasi dari waktu pemrosesan diurutkan dari besar ke kecil	Pilih posisi dengan *balanced machine utilization* yang lebih baik
NEHKK2	Urutkan dari besar ke kecil berdasarkan jumlah pembobotan waktu pemrosesan (c_j). c_j = min (a_i, b_i), di mana $a_i = \sum_{k=1}^{m} t_{j,k} + \sum_{h=1}^{s}\left(\frac{h-\frac{3}{4}}{s-\frac{3}{4}} - \varepsilon\right)\left(t_{s+1-h,i} - t_{t+h,i}\right)$ $b_i = \sum_{k=1}^{m} t_{j,k} - \sum_{h=1}^{s}\left(\frac{h-\frac{3}{4}}{s-\frac{3}{4}} - \varepsilon\right)\left(t_{s+1-h,i} - t_{t+h,i}\right)$ $s = [m/2], t = [m/2]$	Pekerjaan x akan dimasukan di awal jika $a_x \le b_x$. Pekerjaan x akan dimasukan di akhir jika $a_x \ge b_x$.

(lanjut)

Tabel I.2 *Priority Rule* dan *Tie-breaking Rule* pada Algoritma NEH (lanjutan)

Algoritma	***Priority Rule***	***Tie-breaking Rule***
NEHFF	Total waktu proses disusun dari yang besar ke kecil	Pilih posisi dengan *front delay* dan *idle-time* yang paling kecil
NEHLJP	Susun pekerjaan dari besar ke kecil berdasarkan jumlah dari $(AVG_i + MAD_i + abs(SKE)^{1/3} + 1/KUR_i^{1/4})$	Posisi yang dipilih berdasarkan rata-rata dan standar deviasi dari *job flow time* yang dilihat dari nilai P paling besar di mana $P = (\bar{f} - std(f))/C_{max}$

(Sumber: Liu, W., Jin, Y., & Price, M., 2016)

Pada Tabel I.2 dapat dilihat bahwa *priority rule* yang dimiliki algoritma NEH kurang baik karena hanya mengurutkan berdasarkan total waktu proses per pekerjaan dari besar ke terkecil. Hal ini juga diperkuat dengan hasil penelitian yang dilakukan oleh Liu et al. (2016) di mana hasil dari *priority rule* algoritma NEH paling besar (lihat pada Tabel I.3 yang berisikan perbandingan *priority rule*).

Tabel I.3 Perbandingan *Priority Rule*

Instance	NEH	PR_D	PR_{KK1}	PR_{KK2}	PR_{LJP}
20|5	2,07	2,51	1,47	1,95	2,15
20|10	4,80	5,38	3,67	6,18	3,11
20|20	4,04	4,37	4,32	2,44	4,09
50|5	0,79	1,36	0,55	0,55	1,16
50|10	3,69	3,66	3,33	3,79	3,08
50|20	3,52	2,03	2,10	4,33	2,14
100|5	0,80	0,45	0,54	0,66	0,70
100|10	2,30	2,06	2,05	1,60	1,72
100|20	3,73	2,05	2,89	3,44	2,33
200|10	1,12	1,10	0,92	0,76	0,57
200|20	3,45	1,66	2,77	2,93	2,53
500|20	2,00	1,34	2,36	0,48	1,49
AVG	2,69	2,33	2,25	2,42	2,09

(Sumber: Liu, W., Jin, Y., & Price, M., 2016)

Oleh karena itu, pada penelitian ini akan dilakukan penggantian metode *priority rule* yang digunakan pada algoritma NEH. *Priority rule* yang baru akan menggunakan algoritma untuk *permutation flowshop* yang lainnya. Berdasarkan hasil penelitian yang dilakukan oleh Gupta dan Chauhan (2015), algoritma *Gupta Chauhan* memberikan hasil yang lebih baik dari pada tiga algoritma lainnya (algoritma *Palmer,* CDS, dan RA). Hal tersebut dapat dibuktikan pada Tabel I.5 di mana hasil dari algoritma *Gupta Chauhan* (yang diberi tanda hijau) paling baik. Selain itu, menurut Gupta et al. (2015) algoritma *Gupta Chauhan* merupakan algoritma yang baik dalam memberikan solusi awal atau urutan awal yang dapat

digunakan oleh algoritma lain untuk mencari solusi urutan pekerjaan terbaik. Sehingga algortima *Gupta Chauhan* ini akan digunakan sebagai *priority rule* dalam penelitian ini.

Tabel I.4 Perbandingan Hasil Algoritma *Gupta Chauhan*, Palmer, CDS, dan RA

Instance Size	No. of Instances	Average Gap (%)			
		Proposed Heuristic	Palmer	CDS	RA
20x5	10	**7,7**	10,81	9,57	8,81
20x10	10	**10,61**	15,27	12,81	15,39
20x20	10	**8,76**	16,34	9,39	16,34
50x5	10	**4,09**	5,34	6,38	6,18
50x10	10	**10,96**	13,49	12,43	14,5
50x20	10	**12**	15,46	13,31	18,34
100x5	10	2,88	**2,33**	4,95	3,56
100x10	10	**7,64**	9,09	9,11	10,46
100x20	10	**10,53**	13,44	12,13	15,9
200x10	10	5,32	**5,02**	7,42	6,14
200x20	10	**9,4**	12,16	11,14	11,66
500x20	10	**6,29**	6,76	8,36	7,98
Overall	120	**8**	10,46	9,75	11,27

(Sumber: Gupta, A. & Chauhan, S. R., 2015)

Selain akan merubah *priority rule* yang digunakan, pada penelitian ini juga akan dilakukan penggantian metode *tie-breaking rule.* Hal ini dilakukan karena pada algoritma NEH sendiri tidak ada aturan yang jelas untuk *tie-breaking rule*-nya (dapat dilihat pada Tabel I.2 di mana *tie-breaking rule* dari algoritma NEH adalah memilih urutan pekerjaan yang merupakan urutan pekerjaan pertama yang muncul dengan hasil yang sama). Oleh karena itu, pada penelitian ini akan digunakan *tie-breaking rule* dari hasil pengembangan algoritma NEH yang sudah dilakukan.

Tabel I.5 Perbandingan *Tie-breaking Rule*

Instance	NEH	TB_D	TB_{KK1}	TB_{KK2}	TB_{FF}	TB_{LJP}
20|5	0,31	0,35	0,52	0,53	0,78	0,65
20|10	0,55	0,14	0,55	0,55	0,29	0,22
20|20	0,00	0,10	0,00	0,00	0,10	0,10
50|5	0,10	0,57	0,13	0,16	0,40	0,42
50|10	2,66	3,06	3,11	3,11	1,03	2,67
50|20	0,14	0,18	0,16	0,16	0,13	0,20
100|5	0,23	0,20	0,21	0,22	0,28	0,29
100|10	1,31	0,76	0,77	0,78	1,07	0,59
100|20	1,73	0,84	1,09	1,09	1,12	0,60
200|10	1,04	0,74	0,73	0,60	0,91	0,96
200|20	1,87	2,36	1,77	1,86	2,02	2,57
500|20	1,50	1,79	1,78	2,63	1,25	1,39
AVG	0,95	0,92	0,90	0,97	0,78	0,89

(Sumber: Liu, W., Jin, Y., & Price, M., 2016)

Berdasarkan penelitian yang dilakukan oleh Liu et al. (2016), *tie-breaking rule* yang dapat memberikan performansi yang baik adalah *tie-breaking rule* dari NEHFF. Hal ini dapat dilihat pada Tabel I.6 di mana menunjukan perbandingan *tie-breaking rule* dari beberapa algoritma NEH. Selain itu, pemilihan *tie-breaking rule* dari NEHFF dikarenakan pada *tie-breaking rule* NEHFF urutan pekerjaan yang dipilih dipertimbangkan berdasarkan *idle time* yang dimiliki oleh sebuah urutan pekerjaan. Pemilihan *tie-breaking rule* tersebut dikarenakan menurut Liu et al. (2016) jika melakukan minimasi *makespan* belum tentu *idle time* dari urutan pekerjaan yang terbentuk akan berkurang dan *idle time* merupakan salah satu cara untuk meningkatkan utilisasi sumber daya. Sehingga penggunaan kriteria *idle time* pada *tie-breaking rule* atau penggunaan *tie-breaking rule* NEHFF akan baik dalam mencapai tujuan dari penjadwalan atau meningkatkan performansi algoritma NEH. Dengan begitu, pada penelitian ini akan digunakan *tie-breaking rule* dari NEHFF.

Berdasarkan identifikasi yang telah dilakukan, maka ingin dilakukan pembuatan algoritma baru yang berbasiskan pada algoritma NEH. Di mana algoritma baru yang ditawarkan akan menggunakan algoritma *Gupta Chauhan* untuk *priority rule* yang baru dan algoritma NEHFF untuk *tie-breaking rule* yang baru. Penggabungan ini dilakukan untuk menghasilkan pengembangan algoritma NEH yang baru di mana dapat meminimasi *makespan.* Diharapkan dengan dilakukan kedua hal tersebut dapat memberikan performansi yang lebih baik dalam menyelesaikan PFSP.

Dari hasil identifikasi masalah tersebut, dapat dilakukan perumusan masalah dari penelitian yang akan dilakukan. Pada penelitian ini didapatkan beberapa rumusan masalah, yaitu

1. Bagaimana pengembangan algoritma NEH yang baru dalam menyelesaikan PFSP?
2. Bagaimana hasil evaluasi dari pengembangan algoritma NEH yang baru dibandingkan dengan algoritma pesaing yaitu algoritma *Gupta Chauhan*, *Palmer*, CDS, RA, NEH, dan NEHFF dalam menyelesaikan PFSP?

I.3 Pembatasan Masalah dan Asumsi Penelitian

Berdasarkan hasil identifikasi dan perumusan masalah yang telah dilakukan, maka ditentukan beberapa batasan masalah yang akan digunakan. Batasan masalah yang digunakan pada penelitian ini adalah

1. Penelitian ini hanya dilakukan untuk *permutation flowshop scheduling problem* (PFSP).
2. Pada penelitian ini akan menggunakan masalahan penjadwalan *permutation flowshop* dari Taillard (1993) untuk menguji performansi yang dapat diberikan oleh pengembangan algoritma NEH yang baru.

I.4 Tujuan Penelitian

Berdasarkan identifikasi dan rumusan masalah yang telah dilakukan maka didapatkan tujuan dari penelitian. Tujuan penelitian yang mau dicapai dari penelitian ini adalah

1. Mengembangkan algoritma NEH yang baru dalam menyelesaika PFSP.
2. Mengetahui hasil evaluasi dari pengembangan algoritma NEH yang baru dibandingkan dengan algoritma pesaing yaitu algoritma *Gupta Chauhan*, *Palmer*, CDS, RA, NEH, dan NEHFF dalam menyelesaikan PFSP.

I.5 Manfaat Penelitian

Selain tujuan peneletian, sebuah penelitian pasti memiliki manfaat yang mau diberikan dari penelitian ini. Manfaat dari penelitian ini adalah

1. Menambah pengetahuan dan wawasan mengenai penerapan algoritma dalam menyelesaikan PFSP.
2. Menambah referensi untuk penelitian lain yang berkaitan dengan algoritma dalam menyelesaikan PFSP.

I.6 Metodologi Penelitian

Dalam pengembangan algoritma NEH yang baru untuk menyelesaikan PFSP terdapat metodologi yang digunakan. Metodologi penelitian yang digunakan dapat diuraikan sebagai berikut:

1. Studi Literatur

Pada tahap pertama ini akan dilakukan pencarian dan pengumpulan informasi terkait dengan algoritma yang akan digunakan yaitu algoritma NEH, NEHFF, dan *Gupta Chauhan*. Informasi terkait dikumpulkan melalui jurnal dan buku terkait.

2. Identifikasi dan Perumusan Masalah

Pada tahap ini akan dilakukan identifikasi masalah yaitu permasalahan penjadwalan pada *permutation flowshop* dan penerapan metode heuristik (algoritma NEH, NEHFF, dan *Gupta Chauhan*) untuk menyelesaikan permasalahan penjadwalan pada *permutation flowshop*. Selanjutnya dilakukan perumusan masalah berdasarkan identifikasi yang dilakukan.

3. Batasan Masalah dan Asumsi Penelitian

Setelah dilakukan identifikasi dan perumusan masalah akan dilakukan penentuan batasan masalah dan asumsi penelitian yang akan digunakan. Hal ini dilakukan untuk membantu proses penelitian.

4. Tujuan dan Manfaat Penelitian

Pada tahap ini akan dilakukan penentuan tujuan yang mau dicapai dari peneletian dan juga ditentukan manfaat dari penelitian yang dilakukan.

5. Perancangan Pengembangan Algoritma Baru

Pada tahap ini dilakukan perancangan algoritma baru di mana algoritma baru berbasiskan dari algoritma NEH. Akan tetapi, *priority rule* pada algoritma baru akan menggunakan algoritma dari *Gupta Chauhan* dan *tie-breaking rule* pada algoritma baru akan menggunakan algoritma dari NEHFF.

6. Validasi Algoritma

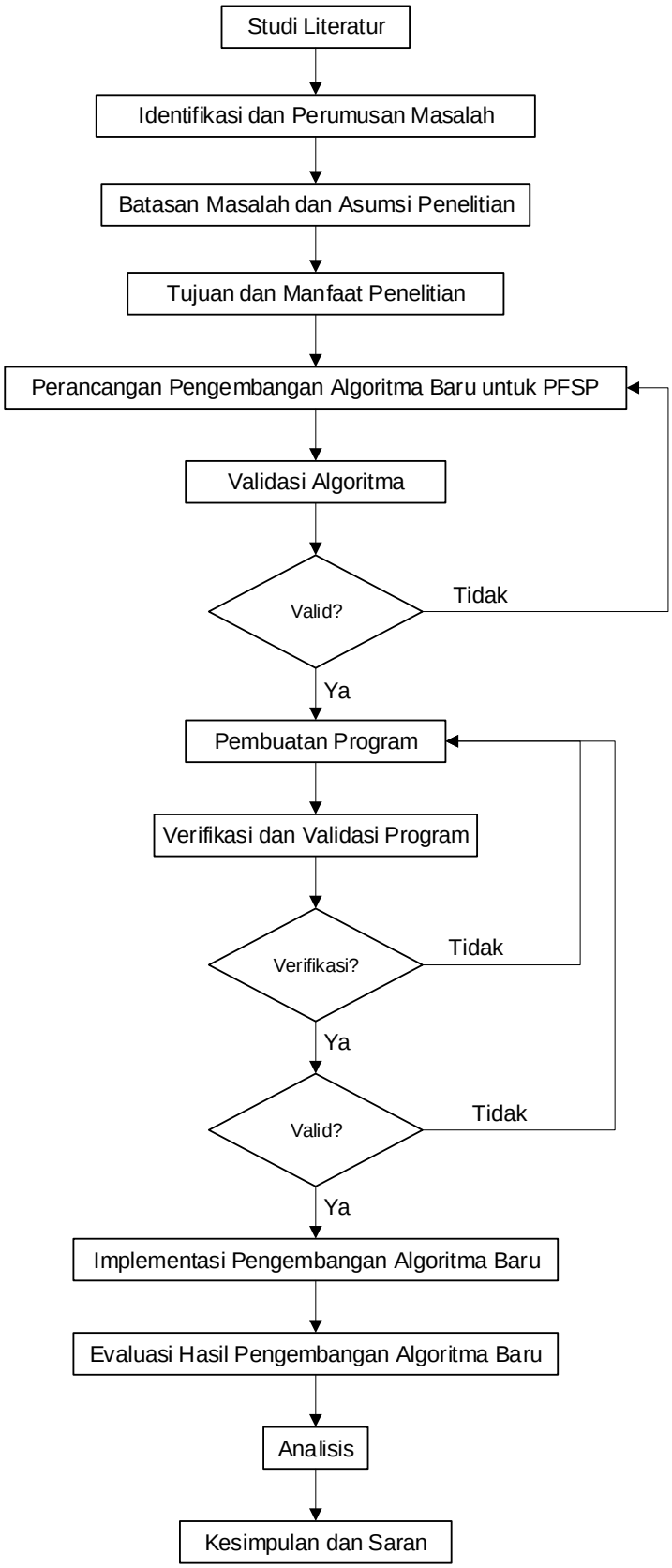
Pada tahap ini akan dilakukan validasi hasil rancangan algoritma dalam menyelesaian PFSP. Validasi algoritma dilakukan untuk mengetahui apakah tahapan pada algoritma baru yang dikembangkan sudah sesuai dengan algoritma NEH, NEHFF, dan *Gupta Chauhan* dalam menyelesaikan PFSP.

7. Pembuatan Program

Pada tahap ini, algoritma yang sudah dirancangan dan tervalidasi akan dibuat programnya. Program dari algoritma baru akan dibuat dengan menggunakan MATLAB.

8. Validasi dan Verifikasi Program

Pada tahap ini akan dilakukan validasi dan verifikasi program. Verifikasi program dilakukan untuk mengetahui apakah program yang dirancang sesuai dengan rancangan algoritma dalam menyelesaikan PFSP. Validasi program dilakukan untuk memastikan memastikan bahwa program yang dirancang sudah berjalan dengan baik sesai dengan rancangan algoritmanya.



Gambar I.1 Metodologi Penelitian

9. Implementasi Pengembangan Algoritma Baru

Program yang telah dibuat akan diterapkan ke dalam *benchmark problem* untuk mengetahui performansi yang dapat diberikan oleh algoritma baru. Data *benchmark problem* yang digunakan adalah data permasalahan penjadwalan untuk *permutation flowshop* dari Taillard.

10. Evaluasi Hasil Pengembangan Algoritma Baru

Hasil dari pengembangan algoritma baru akan dibandingkan dengan hasil dari algoritma pesaing yaitu algoritma *Gupta Chauhan*, *Palmer*, CDS, RA, NEH, dan NEHFF. Hal ini dilakukan untuk mengetahui apakah algoritma baru dapat memberikan hasil yang lebih baik.

11. Analisis

Analisis akan dilakukan terhadap rancangan dari pengembangan algoritma baru untuk menyelesaikan PFSP. Selain itu, analisis juga akan dilakukan terhadap hasil yang diberikan oleh algoritma baru untuk menyelesaikan PFSP dan juga hasil yang diberikan oleh algoritma pesaing.

12. Kesimpulan dan Saran

Pada tahap ini akan dilakukan penarikan kesimpulan yang menjawab tujuan dari penelitian. Selain itu akan diberikan saran yang dapat digunakan untuk penelitian selanjutnya.

I.7 Sistematika Penulisan

Dalam sebuah laporan penelitian diperlukan sistematika penulisan untuk mempermudah pembaca. Sistematika penulisan pada penelitian ini sebagai berikut:

BAB I PENDAHULUAN

Bab I berisikan latar belakang masalah, identifikasi masalah dan rumusan masalah, batasan masalah dan asumsi penelitian, tujuan penelitian, manfaat penelitian, metodologi penelitian, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab II berisikan teori-teori yang terkait didalam penelitian ini seperti *Permutation Flowshop Scheduling Problem* (PFSP) yaitu permasalahan yang akan diselesaikan dan algoritma NEH, NEHFF, dan *Gupta Chauhan* yaitu metode penyelesaian yang digunakan.

BAB III PERANCANGAN DAN IMPLEMENTASI ALGORITMA

Bab III berisikan perancangan dan implementasi dari algoritma baru untuk menyelesaikan PFSP mulai dari perancangan algoritma baru, verifikasi dan validasi algoritma, pembuatan program, verifikasi dan validasi program, dan penerapan algoritma dalam menyelesaikan PFSP.

BAB IV ANALISIS

Bab IV berisikan analisis tahapan yang dilakukan serta evaluasi hasil dari algoritma baru dan juga analisis perbandingan dengan algoritma pesaing.

BAB V KESIMPULAN DAN SARAN

Bab V berisikan kesimpulan dari penelitian yang telah dilakukan serta menjawab tujuan dari penelitian dan juga saran bagi penelitian selanjutnya.