PENERAPAN ALGORITMA *HO-CHANG* DAN *TABU SEARCH* PADA PENJADWALAN *FLOWSHOP*
(Studi Kasus: Industri Jamu Instan Sari Hutani)

SKRIPSI

oleh

Risha Lutfiyan Salam
NIM 071810101088

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2013

PENERAPAN ALGORITMA *HO-CHANG* DAN *TABU SEARCH*
PADA PENJADWALAN *FLOWSHOP*
(Studi Kasus: Industri Jamu Instan Sari Hutani)

SKRIPSI

Diajukan guna melengkapi dan memenuhi salah satu syarat
Untuk menyelesaikan Program Studi Matematika (S1)
Dan mencapai gelar Sarjana Sains

oleh

Risha Lutfiyan Salam
NIM 071810101088

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2013

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ibunda Firdausi dan Ayahanda Agus Mulyanto Adi yang tercinta;
2. guru-guruku sejak taman kanak-kanak sampai dengan perguruan tinggi;
3. Almamater Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

MOTTO

"Allah akan meninggikan orang-orang yang beriman diantara kamu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat."

(Terjemahan *Al-Qur'an* Surat *Al-Mujadalah* ayat 11)[*)]

---

*)Departemen Agama Republik Indonesia.1998. *Al Quran dan terjemahannya* . Semarang: PT Kumudasmoro Grafindo

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Risha Lutfiyan Salam

NIM : 071810101088

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "PENERAPAN ALGORITMA *HO-CHANG* DAN *TABU SEARCH* PADA PENJADWALAN *FLOWSHOP* (Studi Kasus: Industri Jamu Instan Sari Hutani)" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, Februari 2013

Yang menyatakan,

Risha Lutfiyan Salam

NIM. 071810101088

SKRIPSI

PENERAPAN ALGORITMA *HO-CHANG* DAN *TABU SEARCH* PADA PENJADWALAN *FLOWSHOP*

(Studi Kasus: Industri Jamu Instan Sari Hutani)

Oleh

Risha Lutfiyan Salam

NIM. 071810101088

Pembimbing:

Dosen Pembimbing Utama	: Kiswara Agung Santoso, S.Si, M.Kom
Dosen Pembimbing Anggota	: Kusbudiono, S.Si, M.Si

PENGESAHAN

Skripsi berjudul "Penerapan Algoritma *Ho-Chang* dan *Tabu Search* pada Penjadwalan *Flowshop* (Studi Kasus: Industri Jamu Instan Sari Hutani)" telah diuji dan disahkan pada:

Hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim Penguji :

Ketua,	Sekretaris,
Kiswara Agung Santoso, S.Si, M.Kom. NIP. 197209071998031003	Kusbudiono, S.Si, M.Si. NIP. 197704302005011001
Anggota I,	Anggota II,
Prof. Drs. Kusno, DEA., Ph.D. NIP. 196101081986021001	Kosala Dwidja Purnomo, S.Si, M.Si. NIP. 196908281998021001

Mengesahkan
Dekan,

Prof. Drs. Kusno, DEA., Ph.D.
NIP. 196101081986021001

RINGKASAN

Penerapan Algoritma *Ho-Chang* dan *Tabu Search* pada Penjadwalan *Flowshop* (Studi Kasus: Industri Jamu Instan Sari Hutani); Risha Lutfiyan Salam, 071810101088; 2013: 69 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penjadwalan merupakan suatu kegiatan pengalokasian sumber daya yang terbatas untuk mengerjakan sejumlah pekerjaan. Proses penjadwalan timbul jika terdapat keterbatasan sumber daya yang dimiliki, sehingga diperlukan adanya pengaturan sumber-sumber daya yang ada secara efisien. Penjadwalan produksi merupakan kegiatan perencanaan produksi yang terdapat pada perusahaan manufaktur. Adapun tujuan dari penjadwalan produksi umumnya ialah untuk mengoptimalkan dimensi tertentu, yaitu *makespan* (waktu penyelesaian semua tugas atau pekerjaan), keuntungan perusahaan dan waktu tunggu mesin *(machine idletime)*. Penjadwalan *flowshop* adalah salah satu jenis penjadwalan produksi dimana setiap *job* akan melalui setiap mesin dengan urutan yang seragam. Industri jamu instan Sari Hutani merupakan salah satu industri rumahan yang menggunakan pola aliran *flowshop* dalam proses produksinya. Industri tersebut menggunakan 5 buah mesin yakni mesin pencuci (bahan), mesin penggiling (bahan), mesin pemasta, mesin penghancur (pasta), dan mesin pengering, dan menghasilkan 9 jenis produk yaitu pelancar asi, kunci sirih, temulawak, sari jahe, kunyit asam, sari urat, som java, diabetes, kolesterol. Industri Sari Hutani sering kali menambah waktu operasional guna memenuhi permintaan konsumen yang mengakibatkan penambahan biaya produksi. Oleh karena itu, dalam skripsi ini dibahas penyelesaian *flowshop* dengan algoritma *Ho-Chang* dan *Tabu Search* untuk membangun jadwal dengan *makespan* yang optimal serta perbandingan kedua algoritma berdasarkan kompleksitas waktu yang diperlukan.

Penelitian dilakukan melalui beberapa langkah, yaitu mengolah data yang diperoleh menjadi data urutan mesin dan waktu proses kemudian menjadwalkan dengan kedua algoritma. Selanjutnya menghitung kompleksitas waktu dari tiap algoritma, dan membandingkan hasil *makespan* dan kompleksitas waktu yang diperoleh. Yang terakhir adalah menentukan kesimpulan berdasarkan perbandingan sebelumnya.

Penjadwalan yang dilakukan melibatkan 5 buah mesin dan menghasilkan 9 jenis produk jamu instan. Dimana setiap jenis produk jamu instan diproses pada 5 buah mesin yang sama dengan urutan yang seragam. Penjadwalan dengan menggunakan algoritma *Ho-Chang* dan *Tabu Search* menghasilkan nilai *makespan* masing-masing yakni 950 dan 925, sehingga algoritma *Tabu Search* 2,63% lebih efektif jika dibandingkan dengan *makespan* yang dihasilkan dengan algoritma *Ho-Chang.* Artinya, penggunaan algoritma *Tabu Search* lebih efektif jika diaplikasikan pada penjadwalan produksi jamu instan di industri Sari Hutani, karena dapat mengurangi waktu operasional mesin dalam proses produksi sehingga dapat pula mengurangi biaya produksi. Namun berdasarkan kompleksitas waktu yang diperlukan dalam perhitungan, penggunaan algoritma *Tabu Search* dengan $O(n^3m^2)$ membutuhkan waktu yang lebih lama jika dibandingkan dengan menggunakan algoritma *Ho-Chang* dengan $O(n^2m)$. Artinya, penggunaan algoritma *Ho-Chang* lebih efisien jika dibandingkan dengan algoritma *Tabu Search*. Oleh karenanya, dalam skripsi ini disertakan sebuah program Aplikasi Penjadwalan *Flowshop* yang memanfaatkan bahasa pemrograman PHP untuk membantu mempercepat dalam proses perhitungan.

PRAKATA

Puji syukur kehadirat Allah SWT. atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Penerapan Algoritma *Ho-Chang* dan *Tabu Search* pada Penjadwalan *Flowshop* (Studi Kasus: Industri Jamu Instan Sari Hutani)". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata satu (S1) pada jurusan Matematika Fakultas MIPA Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Ibunda Firdausi dan Ayahanda Agus Mulyanto Adi yang tercinta;
2. Kiswara Agung Santoso, S.Si, M.Kom., selaku Dosen Pembimbing Utama dan Kusbudiono, S.Si, M.Si., selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi ini;
3. Prof. Drs. Kusno, DEA., Ph.D., dan Kosala Dwidja Purnomo, S.Si., M.Si., selaku dosen penguji yang telah memberi masukan dalam skripsi ini;
4. Dhika Firdiyan Salam dan Riska Oktavia Wijaya yang tersayang;
5. teman-teman angkatan 2007, Rahma, Sinta, Izzatul dan Dyah. Terima kasih telah menemani dan memberi semangat untuk terus maju menghadapi hari-hari sulit selama masa perkuliahan;
6. Rahayu, Amelia, Edietya, dan Humayra. Terima kasih untuk semangat dan bantuannya;
7. semua pihak yang tidak dapat disebut satu persatu.

Penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini bermanfaat.

Jember, Februari 2013 Penulis

DAFTAR ISI

Halaman

HALAMAN SAMPUL i
HALAMAN JUDUL ii
HALAMAN PERSEMBAHAN iii
HALAMAN MOTTO iv
HALAMAN PERNYATAAN v
HALAMAN PEMBIMBINGAN vi
HALAMAN PENGESAHAN vii
RINGKASAN viii
PRAKATA x
DAFTAR ISI xi
DAFTAR TABEL xiii
DAFTAR GAMBAR xiv
DAFTAR LAMPIRAN xv
BAB 1. PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 3
1.3 Tujuan 3
1.4 Manfaat 4
BAB 2. TINJAUAN PUSTAKA 5
2.1 Definisi Penjadwalan 5
2.2 Penjadwalan *Flowshop* 6
2.3 Grafik Gantt (*Gantt Chart*) 8
2.4 Pengertian Algoritma 9
2.5 Algoritma *Ho-Chang (HC)* 13
2.6 Algoritma *Tabu Search* 16
2.6.1 Konsep Dasar *Tabu Search* 16
2.6.2 Mekanisme *Tabu Search* 17

2.7 Produksi Jamu Instan di Industri Sari Hutani 20
BAB 3. METODE PENELITIAN 24
3.1 Data Penelitian 24
3.2 Langkah-langkah Penyelesaian 25
BAB 4. HASIL DAN PEMBAHASAN 27
4.1 Hasil 27
4.1.1 Penjadwalan dengan Algoritma *Ho-Chang* 28
4.1.2 Penjadwalan dengan Algoritma *Tabu Search* 36
4.1.3 Penjadwalan *Flowshop* dengan Program PHP 37
4.1.4 Perhitungan Kompleksitas Waktu 40
4.2 Pembahasan 63
BAB 5. PENUTUP 66
5.1 Kesimpulan 66
5.2 Saran 67
DAFTAR PUSTAKA 68

DAFTAR TABEL

Halaman

Tabel 2.1 Perbandingan Pertumbuhan $T(n)$ dengan n^2 10
Tabel 2.2 Kelompok Algoritma Berdasarkan Notasi *Big-O* 12
Tabel 3.1 Data Waktu Pembuatan Jamu Instan (menit)..... 25
Tabel 4.1 Perhitungan *Makespan* Solusi Awal 28
Tabel 4.2 Perhitungan *Makespan* Jadwal Baru pada Iterasi Ketiga..... 32
Tabel 4.3 Perhitungan *Makespan* Jadwal Baru pada Iterasi Keempat..... 33
Tabel 4.4 Perhitungan *Makespan* Jadwal Baru pada Iterasi Kelima..... 34
Tabel 4.5 Perhitungan *Makespan* Jadwal Baru pada Iterasi Keenam 35
Tabel 4.6 Solusi Optimal pada *Tabu List*..... 37
Tabel 4.7 Solusi Optimal dari Perhitungan Manual..... 64

DAFTAR GAMBAR

Halaman

Gambar 2.1 Pola Aliran *Flowshop* Murni 6
Gambar 2.2 Pola Aliran *Flowshop* Umum 7
Gambar 2.3 Jenis *Gantt Chart* 8
Gambar 2.4 Ilustrasi *Insertion Move* 18
Gambar 2.5 Ilustrasi *Swap Move* 18
Gambar 2.6 Ilustrasi *Neighborhood Move* 20
Gambar 3.1 Skema Langkah-Langkah Penyelesaian 26
Gambar 4.1 Tampilan Awal Aplikasi Penjadwalan *Flowshop* 38
Gambar 4.2 Tampilan Menu "*Start Input*" 38
Gambar 4.3 Tampilan Tabel pada Input Data 39
Gambar 4.4 Tampilan *Gantt Chart* Hasil Akhir Perhitungan 40
Gambar 4.5 *Flowchart* Solusi Awal Algoritma 42
Gambar 4.6 *Flowchart* Algoritma *Ho-Chang* 49
Gambar 4.7 *Flowchart* Algoritma *Tabu Search* 58

DAFTAR LAMPIRAN

Halaman

A. Hasil Perhitungan *Overall Revised Gaps* Algoritma *Ho-Chang* 70

B. Hasil Pertukaran *Job* Untuk Tiap Iterasi pada Algoritma *Tabu Search* 73

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Permasalahan optimasi merupakan salah satu permasalahan klasik yang sering ditemui di berbagai bidang dalam kehidupan sehari-hari, misalnya dalam bidang industri, transportasi dan perdagangan. Untuk menyelesaikan permasalahan tersebut diperlukan suatu perencanaan yang baik. Suatu perencanaan yang baik apabila dapat menghasilkan keuntungan yang tinggi, meminimalkan biaya dan mampu mengefisienkan sumber daya yang tersedia. Untuk itu, diperlukan suatu proses penjadwalan.

Penjadwalan merupakan suatu kegiatan pengalokasian sumber daya yang terbatas untuk mengerjakan sejumlah pekerjaan. Proses penjadwalan timbul jika terdapat keterbatasan sumber daya yang dimiliki, sehingga diperlukan adanya pengaturan sumber-sumber daya yang ada secara efisien. Permasalahan penjadwalan biasanya berkaitan dengan pengurutan pembuatan atau pengerjaan produk secara menyeluruh terhadap sejumlah mesin yang tersedia. Penjadwalan jenis ini biasa disebut dengan penjadwalan produksi.

Penjadwalan produksi merupakan kegiatan perencanaan produksi yang terdapat pada perusahaan manufaktur. Penjadwalan produksi melibatkan n pekerjaan *(job)* dan m mesin dalam proses produksinya, dimana setiap *job* mengandung informasi tentang jenis produk dan jumlah pesanan. Adapun tujuan dari penjadwalan produksi umumnya untuk mengoptimalkan dimensi tertentu, yaitu *makespan* (waktu penyelesaian semua tugas atau pekerjaan), keuntungan perusahaan dan waktu tunggu mesin *(machine idletime)*. Menurut Ginting (2006), penjadwalan produksi dibedakan menjadi penjadwalan produksi *flowshop* dan penjadwalan produksi *jobshop*. Penjadwalan *flowshop* adalah salah satu jenis penjadwalan produksi dimana setiap *job* akan melalui setiap mesin dengan urutan yang seragam. Sedangkan penjadwalan

jobshop adalah jenis penjadwalan produksi dimana setiap *job* akan diproses tepat satu kali pada tiap mesin dengan urutan yang berbeda sesuai dengan jenis pekerjaannya.

Penjadwalan *flowshop* sering kali dijumpai dalam permasalahan penjadwalan produksi barang dengan jenis barang yang jumlahnya banyak pada sebuah pabrik atau perusahaan, seperti penjadwalan produksi roti pada perusahaan roti, penjadwalan produksi pakaian pada perusahaan konveksi dan penjadwalan produksi jamu instan pada perusahaan jamu instan.

Terdapat dua macam teknik penyelesaian dalam permasalahan penjadwalan *flowshop*, yaitu teknik eksak dan teknik heuristik. Namun, dewasa ini penggunaan teknik eksak untuk kasus yang lebih rumit dirasa sangat tidak efisien. Hal ini dikarenakan penggunaan teknik eksak tidak menjamin solusi atau jadwal yang diperoleh adalah optimal, selain itu juga membutuhkan waktu komputasi yang lebih lama (Bondal, 2008). Adapun teknik heuristik yang akan digunakan untuk menyelesaikan permasalahan penjadwalan produksi *flowshop* kali ini adalah dengan menggunakan algoritma *Ho-Chang (HC)* dan algoritma *Tabu Search*.

Algoritma *HC* mencoba untuk meminimalisasi tidak hanya nilai *makespan*, tetapi juga *machine idletime* (Soetanto, Tessa V.1999). Sedangkan, ide dari algoritma *Tabu Search* adalah melakukan prosedur pengulangan metode pencarian daerah sekitar untuk menemukan suatu solusi yang optimal (Glover. 1998). Inilah yang menjadi bahan pertimbangan penulis, sehingga memilih untuk menganalisa performa dari kedua algoritma tersebut berdasarkan nilai *makespan* dan efisiensi algoritma untuk diterapkan dalam penjadwalan produksi jamu instan Sari Hutani.

Industri jamu instan Sari Hutani terletak di Desa Curahnongko Kecamatan Tempurejo Kabupaten Jember. Terdapat lima mesin yang digunakan dalam proses produksinya dengan sembilan pekerjaan *(job)* yaitu banyaknya jenis jamu instan yang dihasilkan, dengan setiap pekerjaan diproses tepat satu kali pada setiap mesin dengan urutan yang sama. Dalam proses produksinya, industri jamu instan Sari Hutani tidak memiliki jadwal yang tetap atau paten, yakni memproduksi jamu instan berdasarkan jumlah persediaan yang ada di tempat penyimpanan. Sedangkan banyaknya

permintaan untuk tiap jenis jenis jamu pada periode waktu tertentu tidak sama, artinya jumlah produksi tidak dapat memenuhi semua permintaan konsumen. Oleh karenanya, sering kali industri jamu instan Sari Hutani menambah waktu operasional untuk memenuhi permintaan konsumen, yang mengakibatkan penambahan biaya produksi. Dengan pertimbangan tersebut, diharapkan dengan menjadwalkan produksi menggunakan kedua algoritma tersebut diperoleh jadwal yang optimal dalam proses produksi jamu instan sehingga tidak diperlukan penambahan waktu operasional.

1.2 Rumusan Masalah

Permasalahan yang akan dibahas dalam skripsi ini adalah:

a. Bagaimana menyelesaikan permasalahan penjadwalan produksi *flowshop* pada industri jamu instan Sari Hutani dengan algoritma *HC* dan algoritma *Tabu Search*.
b. Manakah algoritma yang baik diantara algoritma *HC* dan algoritma *Tabu Search* jika ditinjau dari nilai *makespan* maupun efisiensi algoritma yang diterapkan pada industri jamu instan Sari Hutani.

1.3 Tujuan

Adapun tujuan dari penulisan skipsi ini adalah:

a. Menerapkan algoritma *HC* dan algoritma *Tabu Search* dalam menyelesaikan permasalahan penjadwalan produksi *flowshop* pada industri jamu instan Sari Hutani guna memperoleh penjadwalan yang meminimumkan *makespan.*
b. Membandingkan kedua algoritma yang diterapkan pada industri jamu instan Sari Hutani berdasarkan nilai *makespan* dan efisiensi algoritma.

1.4 Manfaat

Manfaat yang diperoleh dari penulisan skripsi ini adalah:

a. Mendapatkan penjadwalan produksi jamu instan pada industri jamu instan Sari Hutani yang meminimumkan *makespan* dengan menggunakan *HC* dan algoritma *Tabu Search*.
b. Mendapatkan pemahaman mengenai perbandingan dari kedua algoritma tersebut.

BAB 2. TINJAUAN PUSTAKA

2.1 Definisi Penjadwalan

Penjadwalan adalah pengurutan pembuatan produk secara menyeluruh yang dikerjakan oleh beberapa buah mesin (Conway *et al*, 1967). Sedangkan menurut Baker (1974), penjadwalan didefinisikan sebagai proses pengalokasian sumber daya untuk memilih sekumpulan tugas dalam jangka waktu tertentu. Dalam hal ini sumber daya yang dimaksud tidak hanya berupa mesin yang digunakan namun juga mencakup tenaga kerja yang diperlukan untuk mengoperasikan mesin.

Secara umum, menurut Baker (1974) tujuan dari penjadwalan ialah sebagai berikut:

a. mengurangi persediaan barang setengah jadi dengan cara mengurangi jumlah pekerjaan yang menunggu dalam antrian mesin yang sedang sibuk;
b. meningkatkan produktivitas mesin dengan mengurangi waktu menganggur;
c. mengurangi keterlambatan karena telah melampaui batas waktu;
d. pemenuhan waktu dimana suatu produk harus selesai diproses atau diproduksi *(due date).*

Dengan demikian masalah penjadwalan senantiasa melibatkan pengerjaan sejumlah komponen yang sering disebut dengan istilah *job*. *Job* sendiri masih merupakan komposisi dari sejumlah elemen-elemen dasar yang disebut dengan aktivitas atau operasi. Tiap aktivitas atau operasi ini membutuhkan alokasi atau sumber daya tertentu selama periode waktu tertentu yang sering disebut dengan waktu proses. Selain itu, sumber daya yang dimaksud juga meliputi elemen-elemen lain seperti mesin, transportasi, waktu, dsb (Ginting, 2009).

Penjadwalan secara garis besar dapat dibedakan dalam penjadwalan untuk *jobshop* dan *flowshop*. Permasalahan yang membedakan antara *jobshop* dan *flowshop* adalah pola aliran kerja yang tidak memiliki tahapan-tahapan proses yang sama. Untuk dapat melakukan penjadwalan dengan baik maka waktu proses kerja tiap

mesin serta jenis pekerjaannya perlu diketahui, waktu tersebut dapat diperoleh melalui pengukuran waktu kerja, jenis serta jumlah pekerjaan diperoleh dengan melakukan pengamatan dari operator pada bagian tertentu. Setelah mengetahui jenis serta waktu kerja tiap mesin yang akan dijadwalkan maka proses penjadwalan baru dapat dilakukan (Nisa, Tanpa Tahun).

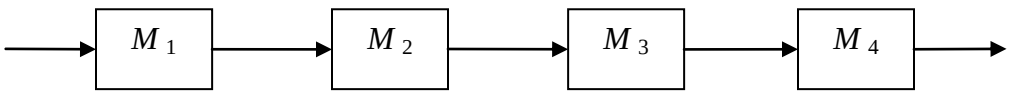
2.2 Penjadwalan *Flowshop*

Penjadwalan *flowshop* merupakan model penjadwalan dimana *job-job* yang akan diproses seluruhnya mengalir pada arah atau jalur produk yang sama. Dengan kata lain, *job-job* yang memiliki urutan kerja yang sama. Umumnya, pada sistem produksi yang bersifat *flowshop*, terdiri dari beberapa mesin (m) dan mempunyai sejumlah *job* yang harus dikerjakan (n) serta waktu proses per unit *job* i pada mesin j, t_{ij} (untuk $i = 1, \dots, n$; $j = 1, \dots, m$). Penjadwalan *flowshop* sering kali diselesaikan dengan mengembangkan permutasi urutan *job* yang akan diurutkan. *Job* bersifat *independent* secara serempak tersedia pada waktu nol, dan urutan mesin dari semua pekerjaan sama. Masing-masing *job* memiliki waktu proses pada masing-masing mesin. Tujuan penjadwalan pada umumnya adalah menemukan suatu urutan *job* yang bertujuan untuk meminimumkan *makespan* (Ginting, 2006).

Penjadwalan *flowshop* dicirikan oleh adanya aliran kerja yang satu arah dan tertentu. Pada dasarnya ada dua macam pola *flowshop* yaitu (Nisa, Tanpa Tahun):

a. *Flowshop* murni *(pure flowshop)*

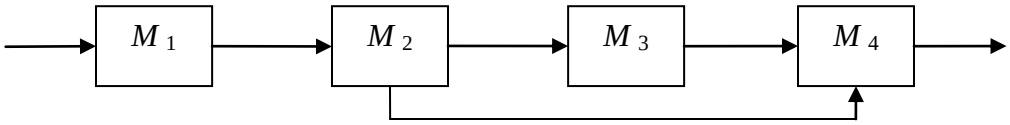
   Kondisi dimana sebuah *job* diharuskan menjalani satu kali proses untuk tiap-tiap tahapan proses. Misalnya, masing-masing *job* melalui mesin 1, kemudian mesin 2, mesin 3 dan seterusnya sampai dengan mesin pada proses yang paling akhir.



Gambar 2.1 Pola Aliran *Flowshop* Murni

b. *Flowshop* umum *(general flowshop)*

Kondisi dimana sebuah *job* boleh melalui seluruh mesin produksi, dimana mulai awal sampai dengan yang terakhir. Dan selain itu sebuah *job* boleh melalui beberapa mesin tertentu, yang mana mesin tersebut masih berdekatan dengan mesin-mesin lainnya dan masih satu arah lintasannya. Berikut ini contoh sistem produksi dengan pola *flowshop* umum:



Gambar 2.2 Pola Aliran *Flowshop* Umum

Masalah yang timbul dari penjadwalan sejumlah pekerjaan yang akan dikerjakan pada sebuah mesin adalah menentukan pekerjaan mana yang pertama kali diproses, kedua, ketiga dan seterusnya. Untuk itu, digunakan beberapa metode eksak dalam menentukan keputusan. Namun jika mesin yang digunakan lebih dari 3, tidak ada algoritma eksak yang dapat menemukan solusi optimalnya. Sekalipun ada, waktu yang dibutuhkan untuk menyelesaikan permasalahan ini secara eksak membutuhkan waktu yang sangat lama. Sehingga untuk mengatasinya ialah dengan menggunakan metode heuristik yang efisien untuk menemukan solusi yang cukup baik (Ravetti, 2006).

Metode heuristik adalah metode yang mulai dari sebuah atau sekumpulan solusi awal, kemudian melakukan pencarian terhadap solusi yang lebih baik hingga mendekati solusi optimal. Metode heuristik baik digunakan untuk menyelesaikan masalah optimasi kombinatorial yang rumit, ketika algoritma eksak sudah tidak mampu untuk menyelesaikan atau masalahnya sulit untuk dipahami dan diformulasikan. Adapun kelebihan dari metode heuristik adalah tidak perlu menganalisa semua kemungkinan solusi untuk mendapatkan solusi optimal dan

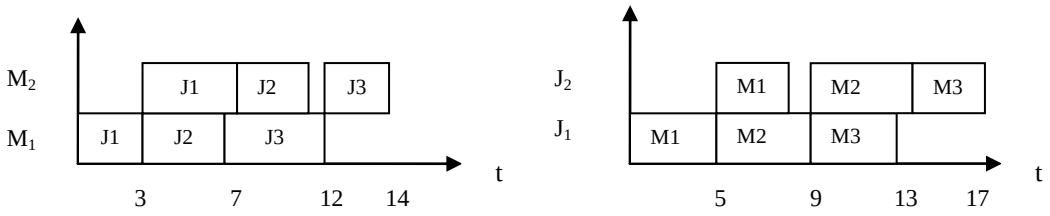
mampu mendapatkan solusi yang mendekati solusi optimal dengan waktu komputasi yang relatif singkat.

2.3 Grafik *Gantt (Gantt Chart)*

Gantt chart pertama kali diperkenalkan oleh Henry Laurence Gantt pada tahun 1916. *Gantt chart* merupakan representasi grafis dari pekerjaan-pekerjaan yang harus diselesaikan dan digambarkan dalam bentuk batang dan analog dengan waktu dan penyelesaian pekerjaan tersebut. Adapun tujuan dibuatnya *gantt chart* yaitu:

a. Menentukan durasi pekerjaan terhadap perkembangan waktu.
b. Perencanaan dan penjadwalan proyek pekerjaan.
c. Pemantauan kemajuan proyek pekerjaan.

Gantt chart terdiri dari 2 jenis yaitu *Machine Oriented Gantt Chart* dan *Job Oriented Gantt Chart* seperti ditunjukkan pada Gambar 2.1 (a) dan Gambar 2.1 (b).



(a) (b)

Gambar 2.3 Jenis *Gantt Chart*

Pada Gambar 2.1 (a), *machine oriented gantt chart* digambarkan dengan sumbu horisontal sebagai durasi atau waktu dan sumbu vertikal sebagai urutan mesin yang digunakan. Sedangkan pada Gambar 2.1 (b), *job oriented gantt chart* digambarkan dengan sumbu horisontal sebagai durasi atau waktu dan sumbu vertikal sebagai urutan *job* yang akan dikerjakan. Dengan kata lain, *machine oriented gantt chart* ialah *gantt chart* yang berorientasi pada mesin dan *job oriented gantt chart* ialah *gantt chart* yang berorienasi pada *job.* Pada prinsipnya, hasil perhitungan dari kedua

jenis *gantt chart* tersebut adalah sama. Pemilihan jenis *gantt chart* yang akan dipakai ditentukan oleh pengguna *gantt chart* itu sendiri (sesuai kebutuhan).

2.4 Pengertian Algoritma

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis dan logis. Kata logis merupakan kata kunci dalam algoritma. Langkah-langkah dalam algoritma harus logis dan harus dapat ditentukan bernilai salah atau benar (Shofwatul'uyun, 2009).

Sedangkan menurut Nurhayati (2009), sebuah algoritma tidak saja harus benar tetapi juga harus mangkus (efisien). Algoritma yang mangkus ialah algoritma yang dapat meminimumkan kebutuhan waktu dan ruang. Kemangkusan (efisiensi) sebuah algoritma dapat digunakan untuk menilai algoritma terbaik. Hal tersebut dapat ditentukan dengan menggunakan:

1. Kompleksitas waktu $T(n)$, merupakan banyaknya operasi yang dilakukan untuk menjalankan sebuah algoritma sebagai fungsi dari ukuran masukan n.
2. Kompleksitas ruang $S(n)$, merupakan besarnya ruang memori yang dibutuhkan algoritma sebagai fungsi dari ukuran masukan n.

Kedua besaran tersebut besifat independen terhadap spesifikasi komputer dan *compiler*. Operasi yang dihitung adalah operasi dasar, yaitu operasi yang mendasari algoritma misal operasi perbandingan elemen pada algoritma, pengurutan/pencarian, penjumlahan dan perkalian. Dengan asumsi bahwa setiap operasi dasar membutuhkan waktu konstan. Sehingga dengan menggunakan besaran kompleksitas waktu atau ruang algoritma, peneliti dapat menentukan laju peningkatan waktu atau ruang yang diperlukan algoritma dengan meningkatnya ukuran masukan n. Dalam penelitian ini, analisis algoritma yang akan digunakan untuk menentukan kemangkusan algoritma adalah kompleksitas waktu $T(n)$ dengan asumsi bahwa setiap operasi dasar yang dilakukan membutuhkan waktu yang konstan. Perhitungan kompleksitas waktu dapat dilakukan dengan menggunakan perhitungan kompleksitas waktu asimptotik.

Kompleksitas waktu asimptotik adalah perkiraan kebutuhan waktu algoritma sejalan dengan meningkatnya nilai n. Pada umumnya, algoritma menghasilkan laju waktu yang semakin lama bila ukuran input n semakin besar. Salah satu cara untuk menghitung kompleksitas waktu dari suatu algoritma adalah dengan menghitung kompleksitas waktu asimptotik dengan menggunakan notasi *Big-O* (O).
Tinjau perbandingan $T(n) = 2n^2 + 6n + 1$ dengan n pada Tabel 2.1 berikut.

Tabel 2.1 Perbandingan Pertumbuhan $T(n)$ dengan n^2

$\boldsymbol{n}$	$\boldsymbol{2n^2 + 6n + 1}$	$\boldsymbol{n^2}$
10	261	100
100	2061	1000
1000	2.006.001	1000.000
10.000	2.000.060.001	1.000.000.000

Untuk n yang besar, pertumbuhan $T(n)$ sebanding dengan n^2. Pada kasus diatas laju pertumbuhan $T(n)$ adalah sama seperti laju pertumbuhan n^2. $T(n)$ bertambah seperti n^2 pada saat n bertambah. Maka dapat dikatakan bahwa $T(n)$ berorde n^2 atau $T(n) = O(n^2)$.

Notasi *Big-O* digunakan untuk menentukan kompleksitas suatu algoritma dengan melihat waktu tempuh algoritma. Selain itu, notasi *Big-O* juga berguna untuk membandingkan beberapa algoritma untuk masalah yang sama sehingga dapat menentukan algoritma terbaik.
Contoh:

Untuk masalah pengurutan memiliki banyak algoritma penyelesaian,

Selection sort, insertion sort → $T(n) = O(n^2)$

Quicksort → $T(n) = O(n \log n)$

Karena $n \log n < n^2$ untuk n yang besar, maka algoritma *Quicksort* dapat dikatakan lebih cepat dibandingkan dengan algoritma *Selection sort* dan *Insertion sort.*

Dari contoh tersebut dapat terlihat bahwa algoritma dengan orde *big-O* yang lebih kecil merupakan algoritma yang lebih efisien. Artinya, semakin kecil orde dari suatu algoritma maka algoritma tersebut akan semakin efisien (Wibowo, 2001).

Menurut definisi:

$T(n) = O(f(n))$ jika terdapat konstanta C dan n_0 sedemikian sehingga

$T(n) \leq C(f(n))$

Artinya, $T(n)$ adalah orde paling besar dari $f(n)$. Misalkan sebuah algoritma memiliki kompleksitas waktu $O(f(n))$, ini berarti untuk n yang besar maka algoritma akan berhenti setelah melakukan operasi dasar paling banyak sebesar konstanta dikalikan $f(n)$. Jadi algoritma membutuhkan konstanta kali $f(n)$ unit waktu (Wibowo, 2001).

Contoh:

$T(n) = 2n^2 + 6n + 1 = O(n^2)$

Penyelesaiannya:

$2n^2 + 6n + 1 = O(n^2)$

karena

$2n^2 + 6n + 1 \leq 2n^2 + 6n^2 + n^2 = 9n^2$ untuk semua $n \geq 1$ ($C = 9$).

Terdapat beberapa kelompok algoritma berdasarkan notasi *Big-O* yang dihasilkan dari perhitungan kompleksitas algoritma, seperti pada tabel 2.2 berikut.

Tabel 2.2 Kelompok Algoritma Berdasarkan Notasi *Big-O*

Kelompok Algoritma	**Nama**
O(1)	Konstan
O(log n)	Logaritmik
O(n)	Linier
O(n log n)	*n* log *n*
O(n^2)	Kuadratik
O(n^3)	Kubik
O(2^n)	Eksponensial
O(n!)	Faktorial

Dan dibawah ini penjelasan dari masing-masing kelompok algoritma:

a. *O(1)*

Kompleksitas *O(1)* berarti waktu pelaksanaan algoritma adalah tetap, tidak bergantung pada ukuran masukan.

b. *O(log n)*

Kompleksitas waktu logaritmik berarti laju pertumbuhan waktunya berjalan lebih lambat daripada pertumbuhan *n*. Algoritma yang termasuk kelompok ini adalah algoritma yang memecahkan persoalan besar dengan mentransformasikan menjadi beberapa persoalan yang lebih kecil yang berukuran sama, misalnya *Binary Search Algorithm*.

c. *O(n)*

Algoritma yang waktu pelaksanaanya linier umumnya terdapat pada kasus yang setiap elemen masukannya dikenai proses yang sama.

d. *O(n log n)*

Waktu pelaksanaan yang *n* log *n* terdapat pada algoritma yang memecahkan persoalan menjadi beberapa persoalan yang lebih kecil, menyelesaikan tiap persoalan secara independen, dan menggabung solusi masing-masing persoalan.

e. $O(n^2)$

Algoritma yang waktu pelaksanaannya kudratik hanya praktis digunakan untuk persoalan yang berukuran kecil. Misalnya pada algoritma pengurutan nilai maksimal, bila n =10 maka waktu pelaksanaan algoritma adalah 100. Bila n dinaikkan dua kali semula maka waktu pelaksanaannya akan menjadi empat kali semula.

f. $O(n^3)$

Seperti halnya algoritma kuadratik, algoritma kubik memproses setiap masukan dalam tiga buah kalang bersarang, misalnya algoritma perkalian matriks.

g. $O(2^n)$

Algoritma yang tergolong kelompok ini mencari solusi persoalan secara *"brute force"*, misalnya pada algoritma mencari sirkuit Hamilton.

h. $O(n!)$

Seperti halnya pada algoritma eksponensial, algoritma jenis ini memproses setiap masukan dan menghubungkannya dengan *n*-1 masukan lainnya, misalnya *Travelling Salesman Problem Algorithm.*

2.5 Algoritma *Ho-Chang* (*HC*)

Pada tahun 1991, Johnny C. Ho dan Yih-Long Chang memperkenalkan sebuah algoritma baru yang bertujuan untuk meminimalisasi *makespan*. Prinsip yang mendasari metode heuristik mereka adalah bahwa minimalisasi jurang-jurang pemisah *(gaps)* antara operasi-operasi yang beriringan akan menghasilkan suatu solusi yang lebih berkualitas. *Gap* didefinisikan sebagai waktu antara berakhirnya *job* ke- i pada mesin ke- j dengan dimulainya *job* ke- i pada mesin ke- $(j + 1)$.

Agar dapat mencapai solusi yang lebih berkualitas, telah dicatat bahwa pasangan *job* dengan *gap-gap* yang paling negatif harus ditempatkan pada bagian akhir jadwal, serta bahwa pasangan *job* yang memiliki *gap-gap* paling positif haruslah ditempatkan pada bagian awal jadwal. Rasionalisasinya adalah bahwa dalam suatu jadwal semacam itu akan terdapat kesempatan-kesempatan yang lebih baik

untuk mengkompensasikan *gap-gap* negatif berkaitan dengan *job-job* belakangan. Teknik pertukaran yang menggunakan ukuran tersebut bertujuan untuk meningkatkan kualitas solusi.

Berikut langkah-langkah pengerjaan algoritma *HC*:

1. Membangkitkan sebuah solusi awal.
2. Menghitung nilai *gap* dengan menggunakan rumus berikut:

$$D_{ij}^{k} = t_{i,k+1} - t_{jk} \qquad (2.1)$$

Dengan :

i, j = 1,2,3, … , n ; $i \neq j$

n = jumlah *job*

k = 1,2,3, … , $(m-1)$

m = jumlah mesin

t = waktu

D_{ij}^{k} = nilai *gap*

Jika *job* i diikuti *job* j pada jadwal, maka nilai D_{ij}^{k} positif mempunyai arti bahwa *job* j harus menunggu pada mesin $(k+1)$ setidaknya selama D_{ij}^{k} sampai *job* i selesai. Sedangkan nilai D_{ij}^{k} negatif mempunyai arti bahwa terdapat *idletime* antara *job* i dan *job* j pada mesin $(k+1)$.

3. Menentukan nilai faktor k (fk) yakni suatu nilai yang diperlukan untuk mengurangi nilai *gap* negatif, dengan menggunakan rumusan sebagai berikut:

$$Faktor\ k\ (fk) = \left\{\left(\frac{(1-0{,}1)}{(m-2)}\right) \times (m-k-1)\right\} + (0{,}1) \qquad (2.2)$$

Dengan:

k = 1,2,3, … , $(m-1)$

m = jumlah mesin

4. Menentukan nilai δ_{ij}^{k} . Jika nilai$D_{ij}^{k} < 0$, maka nilai δ_{ij}^{k} sama dengan nilai dari fk, sedangkan jika nilai D_{ij}^{k} selain itu, maka nilai $\delta_{ij}^{k} = 1$.
5. Menghitung nilai *overall revised gaps* dengan menggunakan persamaan:

$$d_{ij} = \sum_{k=1}^{m-1} D_{ij}^{k} \ \delta_{ij}^{k} \qquad (2.3)$$

Dengan:

i, j = $1,2,3, \dots, n$; $i \neq j$

n = jumlah *job*

k = $1,2,3, \dots, (m-1)$

m = jumlah mesin

δ_{ij}^{k} = fungsi *discount* nilai *gap*

D_{ij}^{k} = nilai *gap*

d_{ij} = nilai *overall revised gaps*

6. Menentukan nilai P_l , set nilai $a = 1$ dan nilai $b = n$ (jumlah *job*).
7. Mencari nilai terbesar yang disebut sebagai X dari $d_{P_a P_l}$ dimana P_a adalah *job* posisi ke a pada solusi awal dan P_l adalah *job* pada posisi ke l dimana $a < l < b$. Kemudian menentukan nilai u (u adalah nilai l yang berkaitan dengan X).
8. Mencari nilai terkecil yang disebut sebagai Y dari $d_{P_l P_b}$ dimana P_b adalah *job* pada posisi ke b pada solusi awal dan P_l adalah *job* pada posisi ke l dimana $a < l < b$. Kemudian menentukan nilai v (v adalah nilai l yang berkaitan dengan Y).
9. Jika (X$<$ 0), (Y $>$ 0) dan (|X| $\leq$ |Y|), maka lanjutkan ke langkah 12.
10. Jika (X$<$ 0), (Y$>$0) dan (|X| $>$ |Y|), maka lanjutkan ke langkah 13.
11. Jika (|X| $>$ |Y|), maka lanjutkan ke langkah 12. Jika tidak, lanjutkan ke langkah 13.
12. Tentukan nilai $a = a + 1$, dan tukar *job* pada posisi ke a dengan *job* pada posisi ke u. Kemudian lanjutkan ke langkah 14.

13. Tentukan nilai $b = b - 1$, dan tukar *job* pada posisi ke b dengan *job* pada posisi ke v.
14. Jika jadwal yang baru memiliki tingkat *performance* yang lebih baik atau sama dengan jadwal yang lama, maka jadwal tersebut akan menjadi solusi awal. Jika tidak, maka jadwal yang lama akan tetap digunakan sebagai solusi awal.
15. Jika $b = a + 2$, maka STOP. Jika tidak, maka kembali ke langkah 7.

2.6 Algoritma *Tabu Search*

Fred Glover (1998) memperkenalkan sebuah teknik heuristik yang disebut *Tabu Search*. Menurut Glover, konsep dasar dari *Tabu Search* merupakan suatu algoritma yang menuntun setiap tahapannya agar dapat menghasilkan kriteria aspirasi yang optimum tanpa terjebak kedalam solusi awal yang ditemukan selama tahapan itu berlangsung. Sehingga maksud dari algoritma ini adalah mencegah terjadinya perulangan dan ditemukannya solusi yang sama pada suatu iterasi yang akan digunakan lagi pada iterasi selanjutnya.

Kelebihan *Tabu Search* terletak pada struktur memori yang fleksibel. Struktur memori itu akan membolehkan pencarian terus dilakukan meskipun solusi yang diperoleh saat ini tidak ada yang lebih baik dari solusi terbaik yang telah diperoleh.

2.6.1 Konsep Dasar *Tabu Search*

Struktur memori dalam *Tabu Search* menggunakan empat prinsip utama yaitu (Berlianty, 2010):

a. *Recency based memory*, yaitu memori yang tetap menjaga struktur terbaik dari solusi awal yang ditemukan selama proses pencarian pada setiap iterasinya, sehingga apabila pada suatu iterasi ditemukan solusi yang lebih baik maka solusi ini akan tetap dipertahankan sampai ditemukannya solusi baru yang lebih baik. Pada metode *Tabu Search* ada kemungkinan terjadi perulangan perhitungan sehingga untuk menghindarinya dibuatlah sebuah struktur yang disebut dengan *Tabu List. Tabu List* berfungsi untuk menyimpan sekumpulan solusi yang telah

dievaluasi (minimal). Pada setiap iterasi, solusi yang akan dievaluasi akan dicocokkan terlebih dahulu dengan *Tabu List*, untuk melihat apakah solusi tersebut sudah ada pada *Tabu List*. Apabila solusi tersebut sudah ada pada *Tabu List*, maka solusi tersebut tidak akan dievaluasi lagi pada iterasi berikutnya.

b. *Frequency*, menyediakan sebuah tipe informasi yang merupakan kumpulan informasi yang telah direkam oleh *Recency based memory*. Sehingga keduanya dapat saling melengkapi untuk membentuk suatu informasi permanen guna mengevaluasi pergerakan *(move)* yang terjadi.

c. *Quality*, adalah kemampuan untuk membedakan solusi terbaik yang dikunjungi selama pencarian atau iterasi berlangsung.

d. *Influence*, mempertimbangkan efek yang terjadi dari pemilihan solusi yang dipilih selama pencarian berlangsung, tidak hanya kualitasnya melainkan juga strukturnya.

2.6.2 Mekanisme *Tabu Search*

Sebagai sebuah algoritma, *Tabu Search* memiliki tahapan dalam penyelesaiannya yakni sebagai berikut (Berlianty, 2010):

1. Membangkitkan sebuah solusi awal
2. Menentukan kriteria aspirasi *(aspiration criteria)*, merupakan sebuah tujuan *(goal)* dalam melakukan perhitungan *tabu search*.
3. Menentukan kriteria terminasi *(stopping criteria)*

   Menurut Panggabean (2005), terdapat beberapa kondisi yang dapat digunakan sebagai kriteria terminasi dalam algoritma *tabu search*, diantaranya adalah:

   a. ditemukannya solusi optimal.
   b. tidak ada lagi solusi baru yang dapat dibangkitkan dari *neighborhood* solusi sekarang, karena semua *move* dalam *neighborhood* tersebut terdapat dalam *tabu list.*

c. jumlah iterasi *I* sama dengan jumlah iterasi maksimum yang ditetapkan di awal.

4. Melakukan *move*

Terdapat beberapa macam *move* yang dapat dipilih dalam proses pencarian solusi terbaik berlangsung, yakni:

a. *Lokal search,* yang terdiri dari dua macam yaitu:

1) *Insertion,* yakni memilih secara acak satu bagian struktur untuk dipindah ke bagian yang lain.

Contoh:

Struktur awal ⇨ | 1 | 2 | 3 | 4 |

Jika dengan proses *random* didapat atribut ke-3, maka struktur dapat berubah menjadi :

1	2	3	4
⇩			
1	3	2	4

Gambar 2.4 Ilustrasi *Insertion Move*

2) *Swap,* yakni memilih secara acak dua bagian struktur untuk selanjutnya ditukar posisinya.

Contoh:

Struktur awal ⇨ | 1 | 2 | 3 | 4 |

Jika dengan proses *random* menghasilkan 1 dan 3, maka struktur dapat berubah menjadi:

1	2	3	4
⇩			
3	2	1	4

Gambar 2.5 Ilustrasi *Swap Move*

b. *Neighborhood Search,* pencarian dengan teknik ini setiap kemungkinan atribut dari struktur dapat dipindah-pindah menggunakan aturan kombinasi.

Contoh:

Struktur awal ⇨

1	2	3	4

Dengan aturan kombinasi 2 dari 4 maka diperoleh struktur sebagai berikut:

1	2	3	4

⇩

2	1	3	4

(a)

1	2	3	4

⇩

3	2	1	4

(b)

1	2	3	4

⇩

4	2	3	1

(c)

1	2	3	4

⇩

1	3	2	4

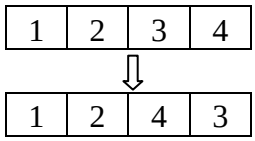
(d)

1	2	3	4

⇩

1	4	3	2

(e)



(f)

Gambar 2.6 Ilustrasi *Neighborhood Move*

5. Menghitung *makespan* dari setiap struktur yang terbentuk, kemudian memilih *makespan* terkecil untuk dimasukkan kedalam *tabu list* untuk menghindari terjadinya *cycling* (mengulang perhitungan).
6. Mengulangi langkah 4 dan 5 hingga tercapai kriteria aspirasi. Jika kriteria terminasi telah dipenuhi maka STOP, artinya nilai *makespan* paling minimum yang berada dalam *tabu list* adalah solusi yang optimum.

2.7 Produksi Jamu Instan di Industri Sari Hutani

Industri Sari Hutani merupakan industri rumahan yang memproduksi beberapa jenis jamu instan yaitu pelancar asi, kunci sirih, temulawak, sari jahe, kunyit asam, sari urat, som java, diabetes, dan kolesterol. Dalam proses produksinya membutuhkan lima buah mesin dengan urutan yang sama untuk tiap jenis produk yakni:

a. Mesin Pencuci (Bahan)

Mesin ini berfungsi untuk mencuci bahan dasar dari jamu instan yang akan diproduksi. Setiap jenis jamu instan yang akan diproduksi membutuhkan waktu yang sama pada proses pencucian yaitu 20 menit untuk 15 kilogram bahan dasar yang akan digunakan.

b. Mesin Penggiling (Bahan)

Mesin ini berfungsi untuk menggiling sekaligus mencampur semua komposisi yang diperlukan dari jamu instan yang akan diproduksi. Setiap jenis jamu instan yang akan diproduksi membutuhkan waktu yang berbeda pada proses ini disesuaikan dengan tekstur dari komposisinya.

c. Mesin Pemasta

Mesin ini berfungsi untuk menghasilkan pasta dari hasil penggilingan pada mesin penggiling bahan. Pasta yang dihasilkan dari mesin ini masih berupa gumpalan-gumpalan yang teksturnya belum sempurna. Proses memasta pada mesin ini membutuhkan waktu yang berbeda untuk setiap jenis jamu instan yang akan diproduksi.

d. Mesin Penghancur (Pasta)

Mesin ini berfungsi untuk menghancurkan gumpalan-gumpalan pasta yang telah dihasilkan oleh mesin pemasta sehingga menghasilkan tekstur pasta yang lebih halus (serbuk). Setiap jenis jamu instan yang akan diproduksi membutuhkan waktu yang sama pada mesin ini yaitu 15 menit.

e. Mesin Pengering

Mesin ini berfungsi untuk menghilangkan kadar air yang masih tersisa pada pasta (serbuk) jamu yang dihasilkan. Waktu yang diperlukan pada proses ini berbeda untuk tiap jenis jamu instan yang akan diproduksi bergantung pada komposisi dari masing-masing produk.

Berikut penjelasan mengenai produk jamu instan yang diproduksi di industri Sari Hutani:

a. Jamu Instan Pelancar Asi

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni membantu melancarkan dan mengurangi bau amis pada ASI. Adapun komposisi dari jamu ini ialah kunyit, temulawak, lempuyang, daun katu, beluntas, jinten hitam, ketumbar, dan gula pasir.

b. Jamu Instan Kunci Sirih

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni sebagai perawatan pasca persalinan dan mengurangi bau badan. Adapun komposisi dari jamu ini ialah kunyit, temulawak, kunci pepet, daun sirih, beluntas dan gula pasir.

c. Jamu Instan Temulawak

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni membantu proses penyembuhan penyakit hepatitis, gangguan fungsi lever dan menambah nafsu makan. Adapun komposisi dari jamu ini ialah temulawak, patikin, daun potro dan gula pasir.

d. Jamu Instan Sari Jahe

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni memperlancar pencernaan, mencegah penggumpalan darah dan menyembuhkan masuk angin. Adapun komposisi dari jamu ini ialah jahe dan gula pasir.

e. Jamu Instan Kunyit Asam

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni membantu menyembuhkan penyakit diabetes mellitus, panas dalam, gangguan fungsi lever dan dismenorhea serta sebagai anti oksidan. Adapun komposisi dari jamu ini ialah kunyit, daun asam dan gula pasir.

f. Jamu Instan Sari Urat

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni mengurangi kadar asam urat dan membantu menghilangkan rasa nyeri ada persendian. Adapun komposisi dari jamu ini ialah temulawak, jahe, lengkuas, jinten hitam, merica hitam, helbeh dan gula pasir.

g. Jamu Instan Som Java

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni sebagai anti oksidan. Adapun komposisi dari jamu ini ialah temulawak, jahe, lengkuas, jinten hitam, merica hitam, daun salam, serai dan gula pasir.

h. Jamu Instan Diabetes

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni membantu menurunkan kadar gula dalam darah pada penderita diabetes mellitus. Adapun komposisi dari jamu ini ialah kunyit, temulawak, bangle, daun bungur, bawang putih, adas dan sambiloto.

i. Jamu Instan Kolesterol

Produk jamu ini berupa serbuk jamu instan yang mempunyai khasiat yakni membantu menurunkan kadar kolesterol dalam tubuh. Adapun komposisi dari jamu ini ialah daun murbai, temulawak, jahe, lengkuas, jinten hitam, merica hitam, daun salam, serai dan gula pasir.

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang digunakan dalam tugas akhir ini merupakan data primer sejumlah lima mesin, sembilan pekerjaan dan waktu pemrosesan tiap pekerjaan pada tiap mesin di industri rumahan pembuat jamu instan Sari Hutani. Industri Sari Hutani yang terletak di kecamatan Tempurejo desa Curahnongko kabupaten Jember ini memiliki lima buah mesin yakni mesin pencuci (bahan), mesin penggiling (bahan), mesin pemasta, mesin penghancur (pasta), dan mesin pengering. Terdapat sembilan produk jamu instan yang dihasilkan yaitu pelancar asi, kunci sirih, temulawak, sari jahe, kunyit asam, sari urat, som java, diabetes, kolesterol.

Pengumpulan data yang akan dilakukan dalam penelitian kali ini adalah dengan cara wawancara, yakni bertanya kepada pemilik maupun operator mengenai waktu pemrosesan tiap pekerjaan pada tiap mesin. Data yang diperoleh berupa data waktu proses pembuatan sembilan jenis produk jamu instan pada tiap mesin dalam satuan waktu yaitu menit. Data tersebut disajikan dalam bentuk Tabel 3.1, dengan sembilan jenis jamu instan *(job)* yang disimbolkan sebagai berikut:

J_1 = jamu instan Pelancar Asi

J_2 = jamu instan Kunci Sirih

J_3 = jamu instan Temulawak

J_4 = jamu instan Sari Jahe

J_5 = jamu instan Kunyit Asam

J_6 = jamu instan Sari Urat

J_7 = jamu instan Som Java

J_8 = jamu instan Diabetes

J_9 = jamu instan Kolesterol

Dan untuk lima mesin yang digunakan, simbol yang diberikan adalah sebagai berikut:

M_1 = Mesin Pencuci (Bahan)

M_2 = Mesin Penggiling (Bahan)

M_3 = Mesin Pemasta

M_4 = Mesin Penghancur (Pasta)

M_5 = Mesin Pengering

Tabel 3.1 Data waktu pembuatan jamu instan (menit)

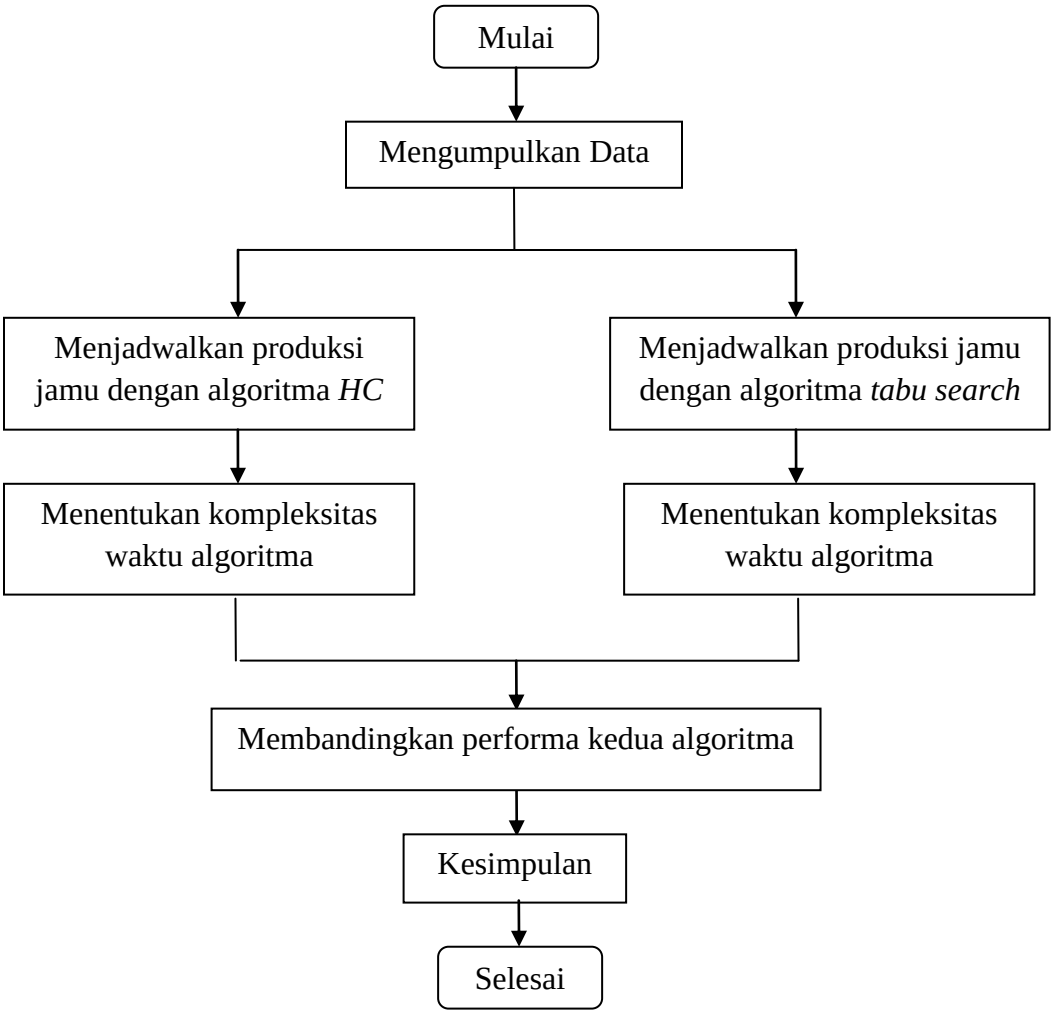
Mesin / ***Job***	M_1	M_2	M_3	M_4	M_5
J_1	20	30	65	15	70
J_2	20	25	45	15	90
J_3	20	30	50	15	90
J_4	20	45	70	15	90
J_5	20	30	60	15	90
J_6	20	45	60	15	100
J_7	20	40	45	15	100
J_8	20	40	50	15	70
J_9	20	30	70	15	120

3.2 Langkah-langkah Penyelesaian

Langkah-langkah yang dilakukan dalam menyelesaikan permasalahan penjadwalan pada industri Sari Hutani adalah sebagai berikut:

1. Mengolah data yang diperoleh menjadi data urutan mesin dan waktu pemrosesan.
2. Menjadwalkan produksi jamu instan berdasarkan data yang diperoleh pada langkah 1 dengan algoritma *HC* dan algoritma *Tabu Search* yang meminimumkan *makespan* secara manual.
3. Menentukan kompleksitas waktu dari masing-masing algoritma.
4. Membandingkan performa dari kedua algoritma, yakni:
   a. Membuat program sesuai algoritma yang digunakan pada langkah 2 menggunakan bahasa pemrograman PHP.
   b. Membandingkan nilai *makespan* dari data yang diperoleh menggunakan program yang telah dibuat.
   c. Membandingkan efisiensi algoritma dengan menghitung kompleksitas waktu dari kedua algoritma.

5. Membuat kesimpulan dari perbandingan sebelumnya.



Gambar 3.1 Skema langkah-langkah penyelesaian

BAB 4. HASIL DAN PEMBAHASAN

Pada bab ini dibahas hasil dari penerapan algoritma *Ho-Chang* dan *Tabu Search* pada penjadwalan jenis *flowshop* pada industri jamu instan Sari Hutani di desa Curahnongko kecamatan Tempurejo kabupaten Jember. Penjadwalan kali ini dilakukan dengan bantuan program PHP. Industri jamu instan Sari Hutani memproduksi sembilan jenis produk jamu yakni pelancar asi, kunci sirih, temulawak, sari jahe, kunyit asam, sari urat, som java, diabetes, kolesterol. Pada proses produksinya, Sari Hutani menggunakan lima buah mesin yaitu mesin pencuci (bahan), mesin penggiling (bahan), mesin pemasta, mesin penghancur (pasta), dan mesin pengering.

4.1 Hasil

Penjadwalan jenis *flowshop* untuk algoritma *Ho-Chang* dan *Tabu Search*, keduanya membutuhkan sebuah solusi awal sebagai acuan dalam perhitungannya. Dalam penelitian kali ini solusi awal yang digunakan merupakan jadwal yang sebelumnya pernah digunakan dalam proses produksi di industri jamu instan Sari Hutani. Berdasarkan data pada Tabel 3.1, maka jadwal yang digunakan sebagai solusi awal memiliki urutan *job* yakni $J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$, dengan perhitungan *makespan*nya pada Tabel 4.1.

Tabel 4.1 Perhitungan *makespan* solusi awal

	M_1		M_2		M_3		M_4		M_5	
	S	**E**	**S**	**E**	**S**	**E**	**S**	**E**	**S**	**E**
J_1	0	20	20	50	50	115	115	130	130	200
J_2	20	40	50	75	115	160	160	175	200	290
J_3	40	60	75	105	160	210	210	225	290	380
J_4	60	80	105	150	210	280	280	295	380	470
J_5	80	100	150	180	280	340	340	355	470	560
J_6	100	120	180	225	340	400	400	415	560	660
J_7	120	140	225	265	400	445	445	460	660	760
J_8	140	160	265	305	445	495	495	510	760	830
J_9	160	180	305	335	495	565	565	580	830	**950**

Keterangan:

S : start atau mulai

E : end atau berhenti

Dari Tabel 4.1 dapat diketahui bahwa nilai *makespan* dari solusi awal yang akan digunakan dalam perhitungan kedua algoritma ialah 950. Artinya dengan jadwal tersebut, industri Sari Hutani membutuhkan waktu 950 menit dalam proses produksinya.

4.1.1 Penjadwalan dengan Algoritma *Ho-Chang*

Berdasarkan tahapan perhitungan algoritma *Ho-Chang* yang telah dijelaskan pada subbab 2.4, maka penjadwalan produksi jamu instan Sari Hutani adalah sebagai berikut:

1. Membangkitkan solusi awal yakni $J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$, dengan *makespan* 950.
2. Melakukan perhitungan dengan menggunakan persamaan (2.1):

   Untuk nilai $i, j = 1,2,3, \dots ,9$,dengan n = *job*

   Untuk nilai $k = 1,2,3, \dots ,4(m - 1)$ dengan m = jumlah mesin.

- $D_{11}^{1} = t_{12} - t_{11} = 30 - 20 = 10$
- $D_{12}^{1} = t_{12} - t_{21} = 30 - 20 = 10$
- $D_{13}^{1} = t_{12} - t_{31} = 30 - 20 = 10$

  ⋮ ⋮ ⋮
- $D_{99}^{4} = t_{95} - t_{94} = 120 - 15 = 105$

3. Menghitung nilai faktor *k* (fk):
   - $f1 = \left\{\left(\frac{(1-0,1)}{(5-2)}\right) \times (5 - 1 - 1)\right\} + (0,1) = 1$
   - $f2 = \left\{\left(\frac{(1-0,1)}{(5-2)}\right) \times (5 - 2 - 1)\right\} + (0,1) = 0,7$
   - $f3 = \left\{\left(\frac{(1-0,1)}{(5-2)}\right) \times (5 - 3 - 1)\right\} + (0,1) = 0,4$
   - $f4 = \left\{\left(\frac{(1-0,1)}{(5-2)}\right) \times (5 - 4 - 1)\right\} + (0,1) = 0,1$
4. Menentukan nilai dari δ_{ij}^{k} . jika nilai$D_{ij}^{k} < 0$, maka nilai δ_{ij}^{k} sama dengan nilai dari faktor (k), sedangkan jika nilai D_{ij}^{k} selain itu, maka nilai $\delta_{ij}^{k} = 1$.
5. Melakukan perhitungan dengan menggunakan persamaan (2.3) pada lampiran A.
6. Menentukan nilai P_l, nilai a dan b
   - Untuk $a = 1$ dan $b = 9$ (iterasi pertama)

     $P_a \rightarrow$ *job* 1

     $P_l \rightarrow$ *job* 2, *job* 3, *job* 4, *j ob* 5, *job* 6, *job* 7, *job* 8

     $d_{p_a p_l} \rightarrow \boldsymbol{d_{12} = 93}$ $d_{16} = 67$

     $d_{13} = 86$ $d_{17} = 78$

     $d_{14} = 63$ $d_{18} = 76$

     $d_{15} = 82$

     Diperoleh nilai X = 93 dan nilai tersebut dimiliki oleh *job* 2 yang menempati posisi ke 2 pada solusi awal sehingga $u = 2$.

     $P_b \rightarrow$ *job* 9

     $P_l \rightarrow$ *job* 2, *job* 3, *job* 4, *j ob* 5, *job* 6, *job* 7, *job* 8

     $d_{p_l p_b} \rightarrow \boldsymbol{d_{29} = 73}$ $d_{69} = 118$

$d_{39} = 83$ $d_{79} = 98$

$d_{49} = 118$ $\boldsymbol{d_{89} = 73}$

$d_{59} = 93$

Karena terdapat dua nilai terkecil yang sama, maka dapat dipilih salah satu, nilai Y = 73 dan nilai tersebut dimiliki oleh *job* 2 yang menempati posisi ke 2 pada solusi awal sehingga $v = 2$.

7. Karena |X| > |Y| maka set nilai $a = a + 1 = 2$ kemudian tukar *job* pada posisi ke a yaitu *job* 2 dengan *job* pada posisi ke u yaitu *job* 2. Karena urutan *job* tidak terjadi perubahan, maka jadwal lama tetap digunakan sebagai solusi awal.
8. Menentukan nilai P_l, nilai a dan b
   - Untuk $a = 2$ dan $b = 9$ (iterasi kedua)

     $P_a \rightarrow$ *job* 2

     $P_l \rightarrow$ *job* 3, *job* 4, *j ob* 5, *job* 6, *job* 7, *job* 8

     $d_{p_a p_l} \rightarrow \boldsymbol{d_{23} = 81}$ $d_{26} = 62$

     $d_{24} = 58$ $d_{27} = 73$

     $d_{25} = 77$ $d_{28} = 71$

     Diperoleh nilai X = 81 dan nilai tersebut dimiliki oleh *job* 3 yang menempati posisi ke 3 pada solusi awal sehingga $u = 3$.

     $P_b \rightarrow$ *job* 9

     $P_l \rightarrow$ *job* 3, *job* 4, *j ob* 5, *job* 6, *job* 7, *job* 8

     $d_{p_l p_b} \rightarrow d_{39} = 83$ $d_{69} = 118$

     $d_{49} = 118$ $d_{79} = 98$

     $d_{59} = 93$ $\boldsymbol{d_{89} = 73}$

     Diperoleh nilai Y = 73 dan nilai tersebut dimiliki oleh *job* 8 yang menempati posisi ke 4 pada solusi awal sehingga $v = 8$.
9. Karena |X| > |Y| maka set nilai $a = a + 1 = 3$ kemudian tukar *job* pada posisi ke a yaitu *job* 3 dengan *job* pada posisi ke u yaitu *job* 3. Oleh karena tidak terjadi

pertukaran urutan *job*, maka jadwal lama yakni $J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$ tetap digunakan sebagai solusi awal perhitungan selanjutnya.

10. Menentukan nilai P_l, nilai a dan b

- Untuk $a = 3$ dan $b = 9$ (iterasi ketiga)

  $P_a \rightarrow$ *job* 3

  $P_l \rightarrow$ *job* 4, *job* 5, *j ob* 6, *job* 7, *job* 8

  $d_{p_a p_l} \rightarrow d_{34} = 68 \qquad d_{37} = 83$

  $\boldsymbol{d_{35} = 87} \qquad d_{38} = 81$

  $d_{36} = 72$

  Diperoleh nilai X = 87 dan nilai tersebut dimiliki oleh *job* 5 yang menempati posisi ke 5 pada solusi awal sehingga $u = 5$.

  $P_b \rightarrow$ *job* 9

  $P_l \rightarrow$ *job* 4, *j ob* 5, *job* 6, *job* 7, *job* 8

  $d_{p_l p_b} \rightarrow d_{49} = 118 \qquad d_{79} = 98$

  $d_{59} = 93 \qquad \boldsymbol{d_{89} = 73}$

  $d_{69} = 118$

  Diperoleh nilai Y = 73 dan nilai tersebut dimiliki oleh *job* 8 yang menempati posisi ke 8 pada solusi awal sehingga $v = 8$.

11. Karena $|X| > |Y|$ maka set nilai $a = a + 1 = 4$ kemudian tukar *job* pada posisi ke a yaitu *job* 4 dengan *job* pada posisi ke u yaitu *job* 5. Sehingga diperoleh jadwal baru yakni $J_1 - J_2 - J_3 - J_5 - J_4 - J_6 - J_7 - J_8 - J_9$, berikut perhitungan *makespan*nya:

Tabel 4.2 Perhitungan *makespan* jadwal baru pada iterasi ketiga

	M1		**M2**		**M3**		**M4**		**M5**	
	S	**E**	**S**	**E**	**S**	**E**	**S**	**E**	**S**	**E**
J_1	0	20	20	50	50	115	115	130	130	200
J_2	20	40	50	75	115	160	160	175	200	290
J_3	40	60	75	105	160	210	210	225	290	380
J_5	60	80	105	135	210	270	270	285	380	470
J_4	80	100	135	180	270	340	340	355	470	560
J_6	100	120	180	225	340	400	400	415	560	660
J_7	120	140	225	265	400	445	445	460	660	760
J_8	140	160	265	305	445	495	495	510	760	830
J_9	160	180	305	335	495	565	565	580	830	**950**

Karena *makespan* jadwal baru sama dengan *makespan* solusi awal maka jadwal baru digunakan sebagai solusi awal.

12. Menentukan nilai P_l, nilai a dan b

- Untuk $a = 4$ dan $b = 9$ (iterasi keempat)

  $P_a \rightarrow$ *job* 5

  $P_l \rightarrow$ *job* 4, *job* 6, *job* 7, *job* 8

  $d_{p_a p_l} \rightarrow d_{54} = 78$ $\qquad$ $\boldsymbol{d_{57} = 93}$

  $d_{56} = 82$ $\qquad$ $d_{58} = 91$

  Diperoleh nilai X = 93 dan nilai tersebut dimiliki oleh *job* 7 yang menempati posisi ke 7 pada solusi awal sehingga $u = 7$.

  $P_b \rightarrow$ *job* 9

  $P_l \rightarrow$ *job* 4, *job* 6, *job* 7, *job* 8

  $d_{p_l p_b} \rightarrow d_{49} = 118$ $\qquad$ $d_{79} = 98$

  $d_{69} = 118$ $\qquad$ $\boldsymbol{d_{89} = 73}$

  Diperoleh nilai Y = 73 dan nilai tersebut dimiliki oleh *job* 8 yang menempati posisi ke 8 pada solusi awal sehingga $v = 8$.

13. Karena |X| > |Y| maka set nilai $a = a + 1 = 5$ kemudian tukar *job* pada posisi ke a yaitu *job* 4 dengan *job* pada posisi ke u yaitu *job* 7. Sehingga diperoleh jadwal baru yakni $J_1 - J_2 - J_3 - J_5 - J_7 - J_6 - J_4 - J_8 - J_9$, berikut perhitungan *makespan*nya:

Tabel 4.3 Perhitungan *makespan* jadwal baru pada iterasi keempat

	M1		**M2**		**M3**		**M4**		**M5**	
	S	**E**	**S**	**E**	**S**	**E**	**S**	**E**	**S**	**E**
J_1	0	20	20	50	50	115	115	130	130	200
J_2	20	40	50	75	115	160	160	175	200	290
J_3	40	60	75	105	160	210	210	225	290	380
J_5	60	80	105	135	210	270	270	285	380	470
J_7	80	100	135	175	270	315	315	330	470	570
J_6	100	120	175	220	315	375	375	390	570	670
J_4	120	140	220	265	375	445	445	460	670	760
J_8	140	160	265	305	445	495	495	510	760	830
J_9	160	180	305	335	495	565	565	580	830	**950**

Karena *makespan* jadwal baru sama dengan *makespan* solusi awal maka jadwal baru digunakan sebagai solusi awal.

14. Menentukan nilai P_l, nilai a dan b

- Untuk $a = 5$ dan $b = 9$ (iterasi kelima)

  $P_a \rightarrow$ *job* 7

  $P_l \rightarrow$ *job* 6, *job* 4 , *job* 8

  $d_{p_a p_l} \rightarrow d_{74} = 83$

  $d_{76} = 87$

  $\boldsymbol{d_{78} = 96}$

  Diperoleh nilai X = 96 dan nilai tersebut dimiliki oleh *job* 8 yang menempati posisi ke 8 pada solusi awal sehingga $u = 8$.

  $P_b \rightarrow$ *job* 9

  $P_l \rightarrow$ *job* 6, *job* 4, *job* 8

$d_{p_l p_b} \rightarrow d_{69} = 118$

$d_{49} = 118$

$\boldsymbol{d_{89} = 73}$

Diperoleh nilai Y = 73 dan nilai tersebut dimiliki oleh *job* 8 yang menempati posisi ke 8 pada solusi awal sehingga $v = 8$.

15. Karena |X| > |Y| maka set nilai $a = a + 1 = 6$ kemudian tukar *job* pada posisi ke a yaitu *job* 6 dengan *job* pada posisi ke u yaitu *job* 8. Sehingga diperoleh jadwal baru yakni $J_1 - J_2 - J_3 - J_5 - J_7 - J_8 - J_4 - J_6 - J_9$, berikut perhitungan *makespan*nya:

Tabel 4.4 Perhitungan *makespan* jadwal baru pada iterasi kelima

	M1		**M2**		**M3**		**M4**		**M5**	
	S	**E**	**S**	**E**	**S**	**E**	**S**	**E**	**S**	**E**
$\boldsymbol{J_1}$	0	20	20	50	50	115	115	130	130	200
$\boldsymbol{J_2}$	20	40	50	75	115	160	160	175	200	290
$\boldsymbol{J_3}$	40	60	75	105	160	210	210	225	290	380
$\boldsymbol{J_5}$	60	80	105	135	210	270	270	285	380	470
$\boldsymbol{J_7}$	80	100	135	175	270	315	315	330	470	570
$\boldsymbol{J_8}$	100	120	175	215	315	365	365	380	570	640
$\boldsymbol{J_4}$	120	140	215	260	365	435	435	450	640	730
$\boldsymbol{J_6}$	140	160	260	305	435	495	495	510	730	830
$\boldsymbol{J_9}$	160	180	305	335	495	565	565	580	830	**950**

Karena *makespan* jadwal baru sama dengan *makespan* solusi awal maka jadwal baru digunakan sebagai solusi awal.

16. Menentukan nilai P_l, nilai a dan b

- Untuk $a = 6$ dan $b = 9$ (iterasi keenam)

  $P_a \rightarrow$ *job* 8

  $P_l \rightarrow$ *job* 4, *job* 6

  $d_{p_a p_l} \rightarrow d_{84} = 58$

  $\boldsymbol{d_{86} = 62}$

Diperoleh nilai X = 62 dan nilai tersebut dimiliki oleh *job* 6 yang menempati posisi ke 8 pada solusi awal sehingga $u = 8$.

$P_b \rightarrow$ *job* 9

$P_l \rightarrow$ *job* 4, *job* 6

$d_{p_l p_b} \rightarrow \boldsymbol{d_{49} = 118}$

$\boldsymbol{d_{69} = 118}$

Karena terdapat dua nilai yang sama, maka dapat dipilih salah satu, nilai Y = 118 dan nilai tersebut dimiliki oleh *job* 4 yang menempati posisi ke 7 pada solusi awal sehingga $v = 7$.

17. Karena X < Y maka set nilai $b = b - 1 = 8$ kemudian tukar *job* pada posisi ke *b* yaitu *job* 6 dengan *job* pada posisi ke v yaitu *job* 4. Sehingga diperoleh jadwal baru yakni $J_1 - J_2 - J_3 - J_5 - J_7 - J_8 - J_6 - J_4 - J_9$, berikut perhitungan *makespan*nya:

Tabel 4.5 Perhitungan *makespan* jadwal baru pada iterasi keenam

	M1		**M2**		**M3**		**M4**		**M5**	
	S	**E**	**S**	**E**	**S**	**E**	**S**	**E**	**S**	**E**
$\boldsymbol{J_1}$	0	20	20	50	50	115	115	130	130	200
$\boldsymbol{J_2}$	20	40	50	75	115	160	160	175	200	290
$\boldsymbol{J_3}$	40	60	75	105	160	210	210	225	290	380
$\boldsymbol{J_5}$	60	80	105	135	210	270	270	285	380	470
$\boldsymbol{J_7}$	80	100	135	175	270	315	315	330	470	570
$\boldsymbol{J_8}$	100	120	175	215	315	365	365	380	570	640
$\boldsymbol{J_6}$	120	140	215	260	365	425	425	440	640	740
$\boldsymbol{J_4}$	140	160	260	305	425	495	495	510	740	830
$\boldsymbol{J_9}$	160	180	305	335	495	565	565	580	830	**950**

18. Menentukan nilai P_l, nilai a dan b

- Untuk $a = 6$ dan $b = 8$

Karena nilai $b = a + 2$ maka STOP.

Perhitungan dengan menggunakan algoritma *Ho-Chang* berhenti pada iterasi keenam dengan solusi optimal $J_1 - J_2 - J_3 - J_5 - J_7 - J_8 - J_6 - J_4 - J_9$, dengan nilai *makespan* 950.

4.1.2 Penjadwalan dengan Algoritma *Tabu Search*

Berdasarkan tahapan perhitungan algoritma *Tabu Search* yang telah dijelaskan pada subbab 2.5, maka penjadwalan produksi jamu instan Sari Hutani adalah sebagai berikut:

1. Membangkitkan solusi awal yakni $J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$, dengan *makespan* 950.
2. Menentukan kriteria aspirasi, yakni pencarian solusi atau urutan *job* yang meminimumkan *makespan.*
3. Menentukan kriteria terminasi, dalam skripsi ini kriteria terminasi yang digunakan ialah point c pada subbab 2.5.2, yakni menentukan jumlah iterasi maksimal sebanyak 6 iterasi.
4. Melakukan *move,* perhitungan dalam skripsi ini menggunakan *neighborhood search* sehingga diperoleh pertukaran *job* pada tiap iterasi seperti pada lampiran B. Dimana setiap solusi paling optimal yang pertama ditemukan pada suatu iterasi, disimpan kedalam *tabu list* yang kemudian akan digunakan sebagai solusi awal untuk melakukan *move* pada iterasi berikutnya.
5. Kemudian ketika kriteria terminasi sudah terpenuhi, yang artinya perhitungan telah mencapai jumlah iterasi maksimum yang telah ditetapkan maka STOP. Kemudian memilih *makespan* paling minimum pada *tabu list* berikut:

Tabel 4.6 Solusi Optimal pada *Tabu List*

Pertukaran	**Urutan *job***	***makespan***
***job* 1 dengan *job* 2**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	**925**
***job* 1 dengan *job* 3**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 4**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 1**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 5**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 4**	$J_2 - J_4 - J_5 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	925

Pada Tabel 4.6 terdapat enam solusi optimal yang telah disimpan dalam *tabu list*, yang memiliki nilai makespan sama yakni 925. Artinya, dari proses perhitungan menggunakan algoritma *Tabu Search* diperoleh nilai *makespan* yang paling optimal yakni 925 menit dan terdapat beberapa atau lebih dari satu solusi optimal. Karenanya, maka solusi optimal dapat dipilih salah satu dari keenam solusi pada *tabu list* yakni $J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$.

4.1.3 Penjadwalan *Flowshop* dengan Program PHP

Dalam skripsi ini, disertakan sebuah Aplikasi Penjadwalan *Flowshop* dengan program yakni menggunakan bahasa pemrograman PHP. Aplikasi tersebut dibuat ialah untuk mempercepat dan mempermudah dalam melakukan perhitungan menggunakan kedua algoritma. Gambar 4.1 merupakan tampilan awal dari program Aplikasi Penjadwalan *Flowshop.* Pada Gambar 4.1, terdapat empat menu utama yang ditampilkan yakni:

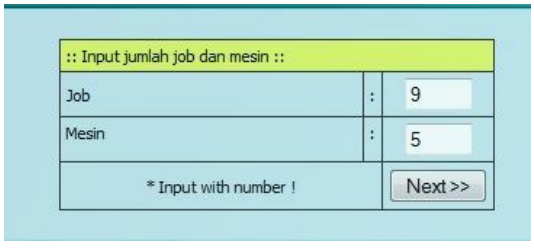
a. *Home,* yaitu tampilan menu awal sebelum melakukan proses *input* data.
b. *Start Input,* yaitu untuk memulai meng-*input* ukuran masalah yang akan diselesaikan dengan program.
c. *Ho-Chang,* yaitu untuk melakukan perhitungan menggunakan algoritma *Ho-Chang* pada data yang telah di*input*.
d. *Tabu Search,* yaitu untuk melakukan perhitungan menggunakan algoritma *Tabu Search* pada data yang telah di*input*.

Gambar 4.1 Tampilan Awal Aplikasi Penjadwalan *Flowshop*

Terdapat beberapa langkah yang harus dilakukan untuk menjalankan program Aplikasi Penjadwalan *Flowshop*, berikut penjelasan dari tiap langkah dalam menjalankan program Aplikasi Penjadwalan *Flowshop*.

1. *Input* Ukuran Masalah

Ukuran masalah penjadwalan *flowshop* dalam skripsi ini adalah 9×5 atau sembilan *job* yang akan diproses pada lima mesin, sehingga jadwal yang terbentuk sebanyak 45 digit jadwal dengan satuan waktu yakni menit. Pengguna harus memilih menu "*Start Input"* untuk memulai meng-*input* ukuran masalah dan dilanjutkan dengan klik tombol "*Next"*untuk memulai meng-*input* data yang akan diselesaikan dengan program seperti pada Gambar 4.2.



Gambar 4.2 Tampilan Menu "*Start Input"*

Program Aplikasi Penjadwalan *Flowshop* ini dibuat dengan memperhatikan tingkat kesulitan algoritma yang digunakan. Oleh karenanya, penulis membatasi

ukuran masalah maksimal yang dapat dieksekusi pada program ini adalah 10×10 atau sepuluh *job* pada sepuluh mesin.

2. *Input* dan *Process* Data

Pada langkah ini, akan muncul tabel *input* data seperti pada Gambar 4.3(a) dan Gambar 4.3(b). Dalam hal ini, pengguna dipersilahkan untuk meng-*input* data yang akan diselesaikan dengan program sesuai dengan ukuran masalah yang telah ditentukan sebelumnya.

(a) (b)

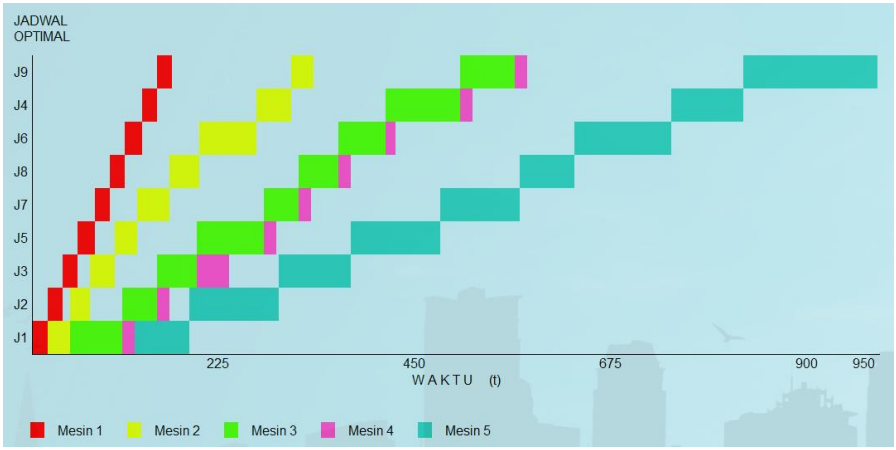
Gambar 4.3 Tampilan Tabel pada *Input* Data

Pada Gambar 4.3(a) terdapat menu pilihan untuk algoritma penyelesaian yang akan digunakan. Ketika algoritma *Ho-Chang* dipilih, pengguna dapat secara langsung meng-*klik* tombol "Lakukan Perhitungan" untuk memproses data yang telah di-*input* sebelumnya. Berbeda dengan ketika algoritma *Tabu Search* dipilih, maka pengguna akan diminta untuk memasukkan jumlah iterasi yang diinginkan terlebih dahulu seperti pada Gambar 4.3(b).

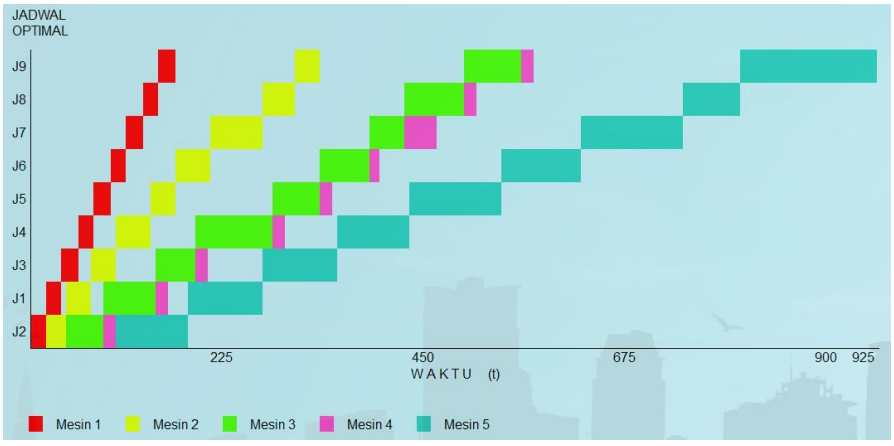
3. *Output* Program

Hasil akhir yang ditampilkan dalam perhitungan menggunakan program ini ialah sebuah jadwal yang meminimumkan nilai *makespan* dari masing-masing algoritma dan gambar *gantt chart*. Gambar 4.4 (a) dan Gambar 4.4 (b) merupakan *gantt chart* dari hasil akhir perhitungan untuk algoritma *Ho-Chang* dan *Tabu Search*, artinya dari data yang telah di-*input* dapat diperoleh jadwal yang meminimumkan

makespan menurut algoritma *Ho-Chang* ialah $J_1 - J_2 - J_3 - J_5 - J_7 - J_8 - J_6 - J_4 - J_9$, sedangkan untuk algoritma *Tabu Search* $J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$. Urutan *job* dari masing-masing algoritma digambarkan dengan sumbu vertikal, dan sumbu horisontal menggambarkan pergerakan waktu. Warna yang digunakan pada *gantt chart* bertujuan untuk memudahkan pengguna untuk membedakan waktu yang diperlukan oleh tiap *job* ketika diproses dalam urutan mesin yang ada.



(a)



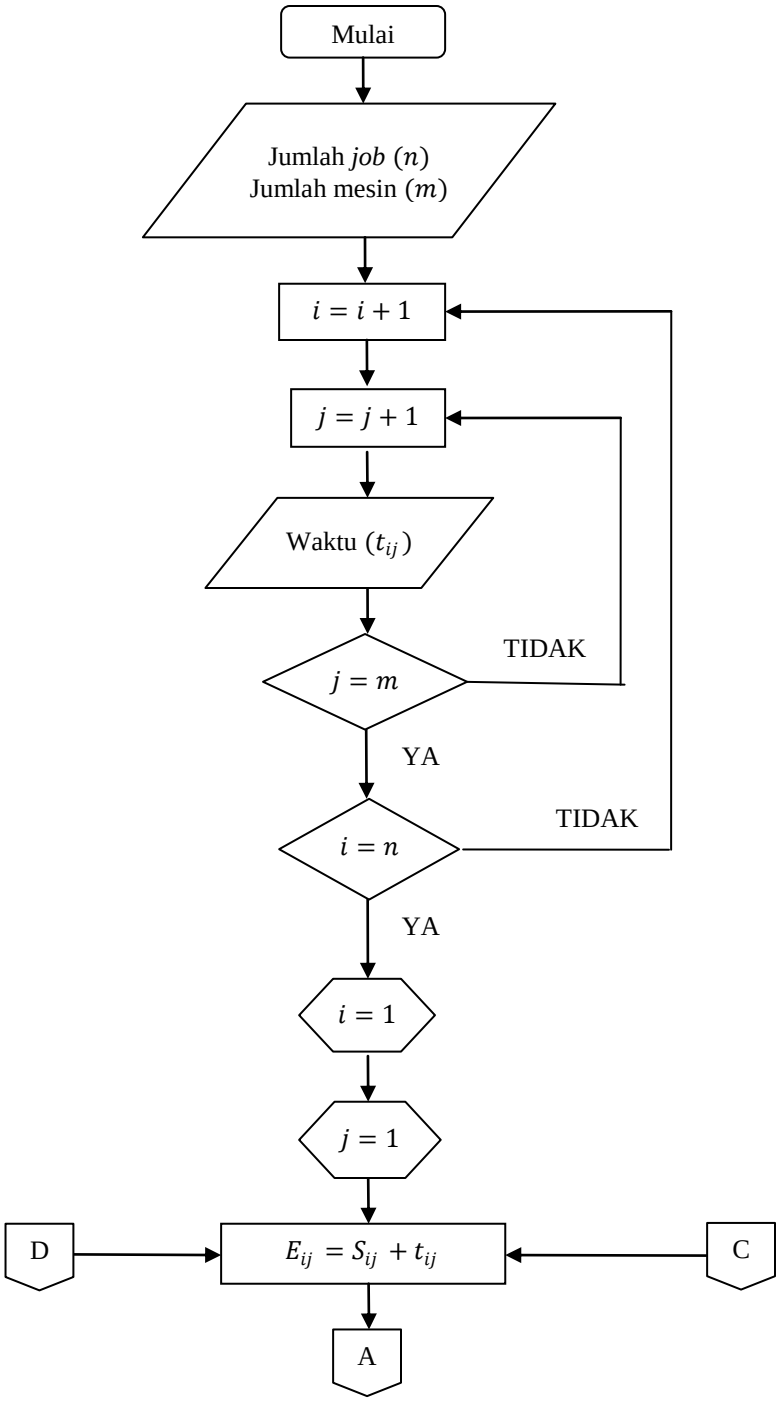
(b)

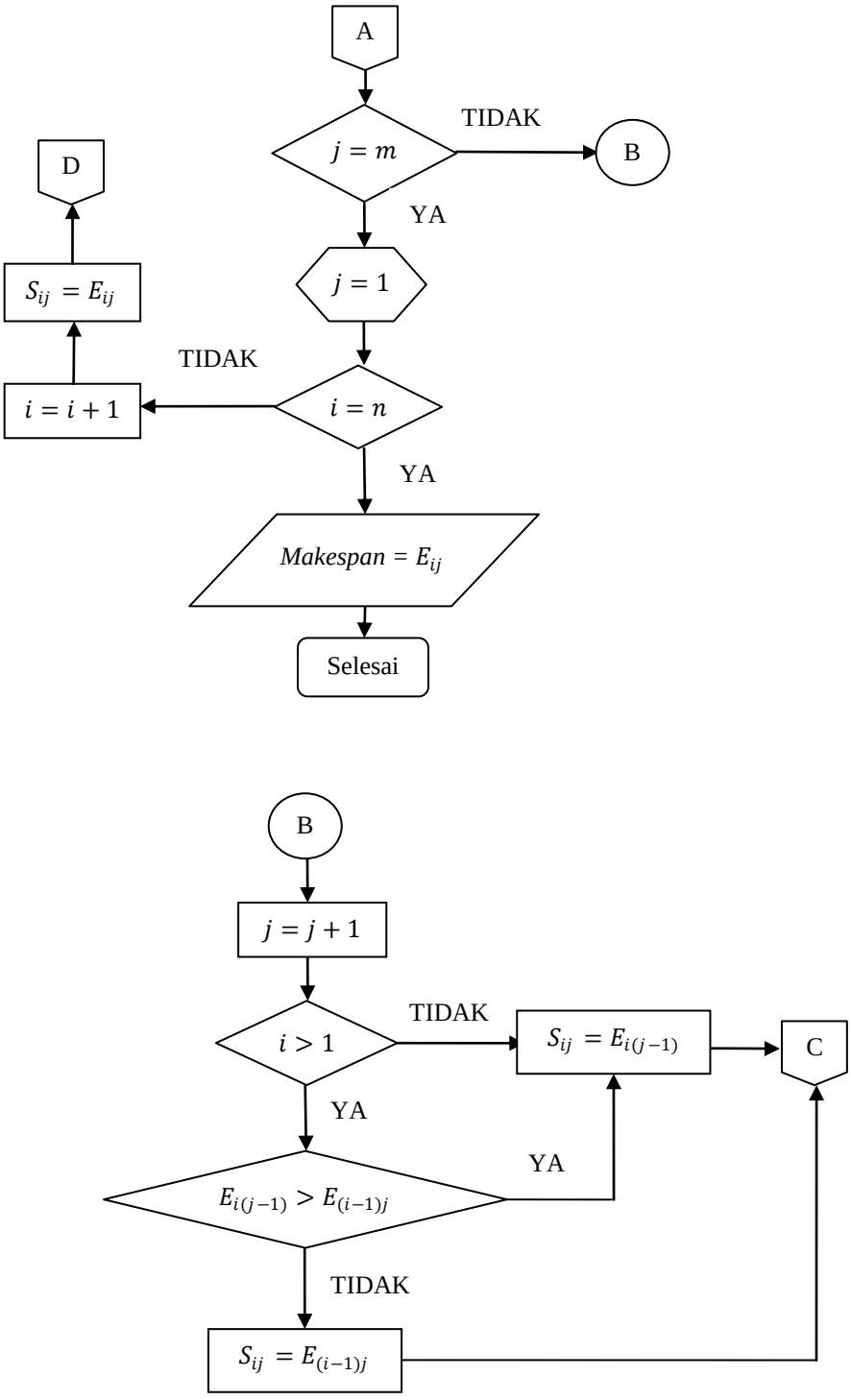
Gambar 4.4 Tampilan *Gantt chart* Hasil Akhir Perhitungan

4.1.4 Perhitungan Kompleksitas Waktu

Untuk menentukan kemangkusan algoritma, penulis menyertakan perhitungan kompleksitas waktu algoritma dengan memperhatikan tahapan komputasi masing-

masing algoritma pada *flowchart*. Oleh karena kedua algoritma sama-sama membutuhkan solusi awal sebagai acuan untuk perhitungan selanjutnya, maka Gambar 4.5 menampilkan tahapan perhitungan solusi awal yang digunakan.





Gambar 4.5 *Flowchart* solusi awal algoritma

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan solusi awal untuk kedua algoritma.

1. Perhitungan untuk *looping* pertama (*input* t_{ij}):

- Untuk $i = 1$,

  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = m kali

- Untuk $i = 2$,

  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = m kali

…

- Untuk $i = n$,

  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = m kali

Jadi, jumlah perhitungannya → $T(n,m)_1 = n \times m = nm$

2. Perhitungan untuk *looping* kedua (hitung E_{ij}):

- Untuk $i = 1$,

  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = m kali

- Untuk $i = 2$,

  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = m kali

…

- Untuk $i = n$,

  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = m kali

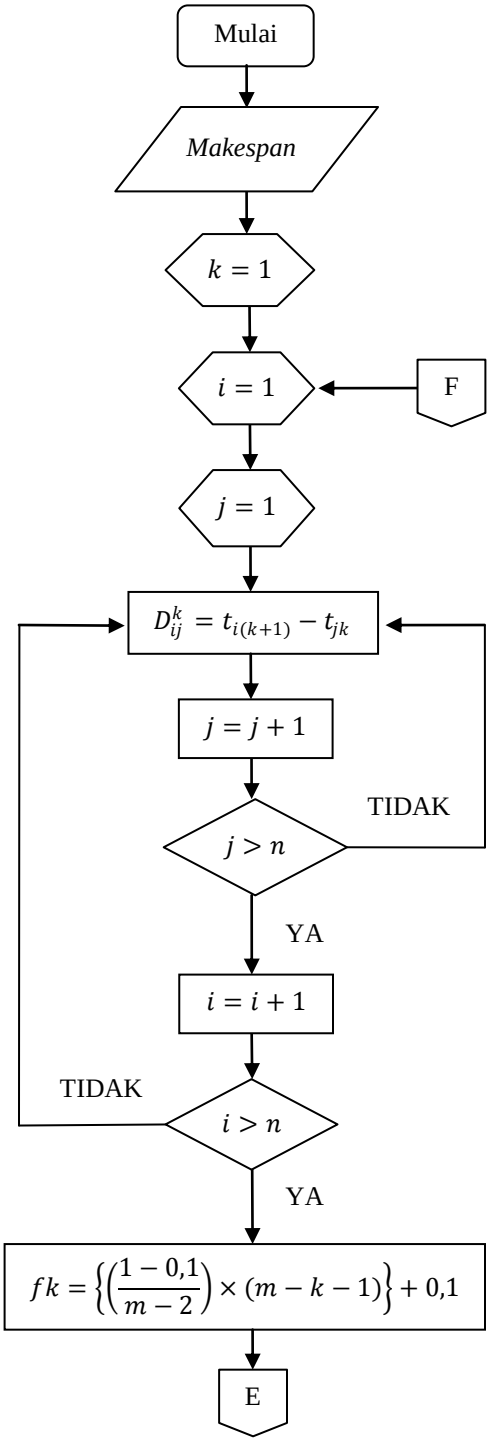
Jadi, jumlah perhitungannya → $T(n,m)_2 = n \times m = nm$

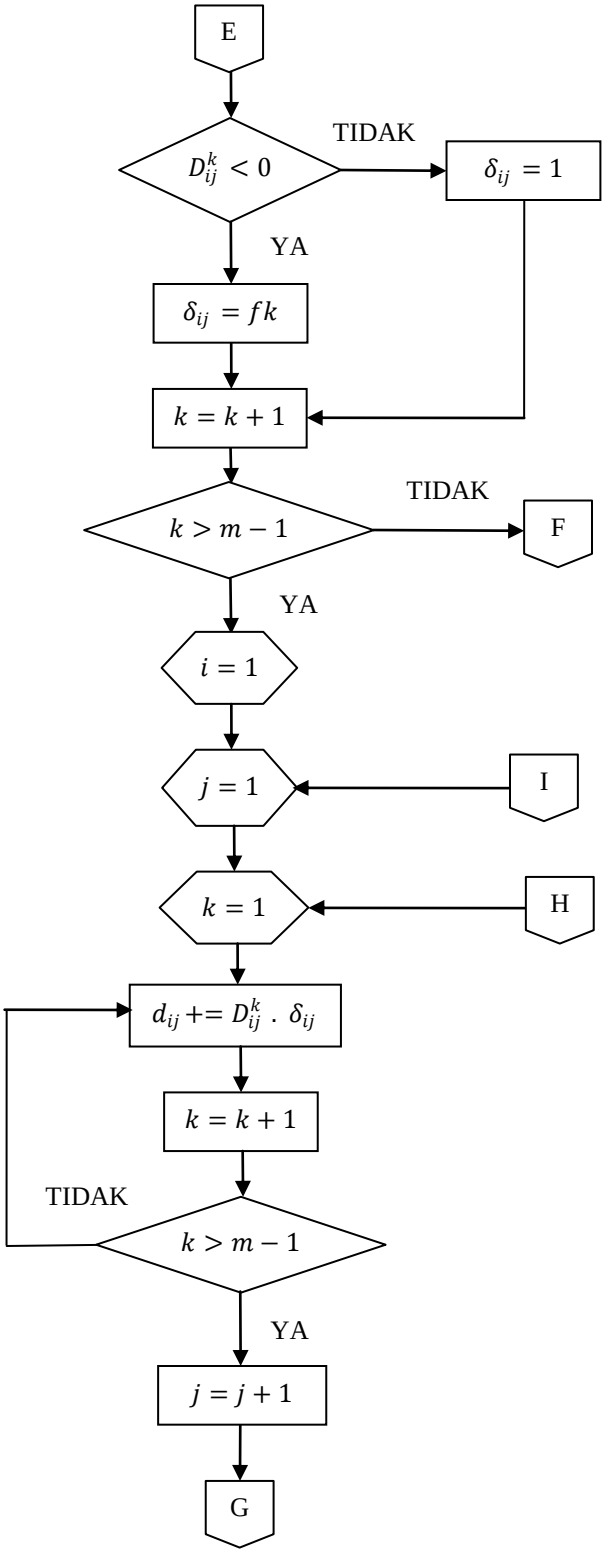
Sehingga total kompleksitas waktu dari tahapan perhitungan solusi awal diperoleh

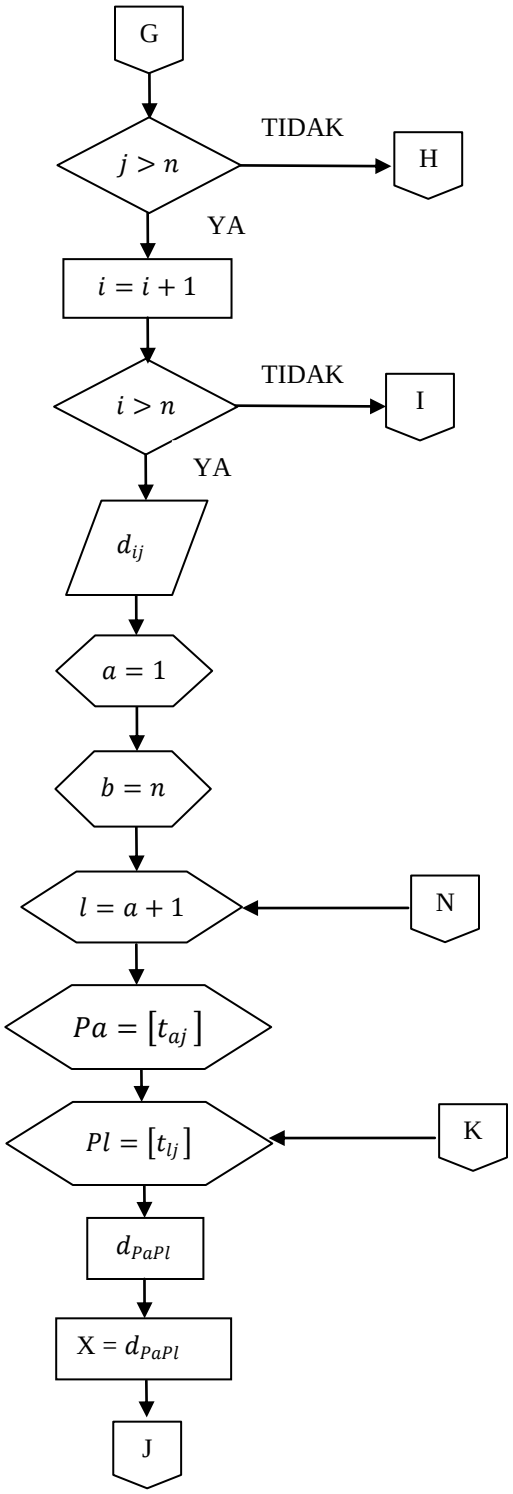
$$T(n,m) = T(n,m)_1 + T(n,m)_2 = nm + nm = 2nm$$

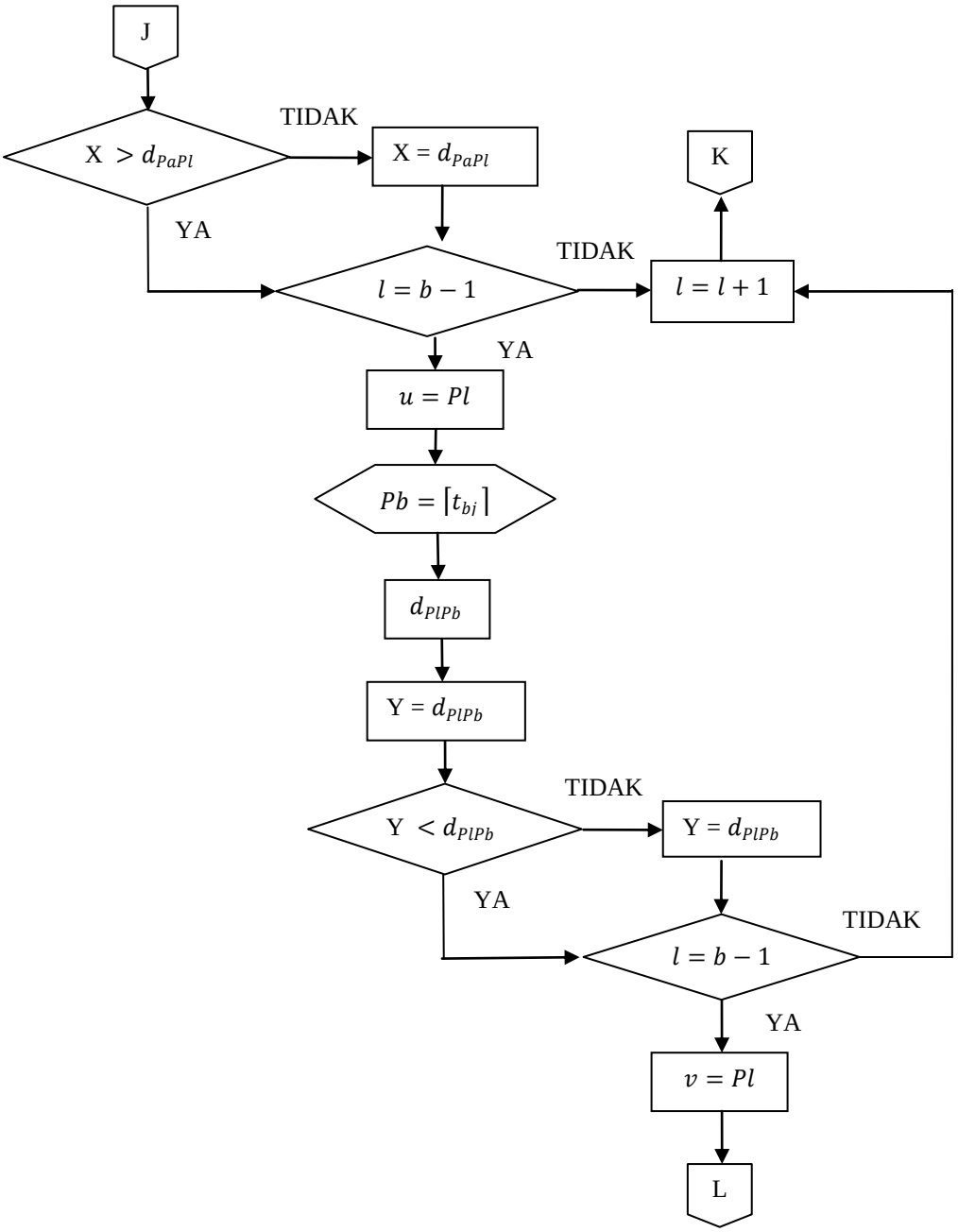
1. Algoritma *Ho-Chang (HC)*

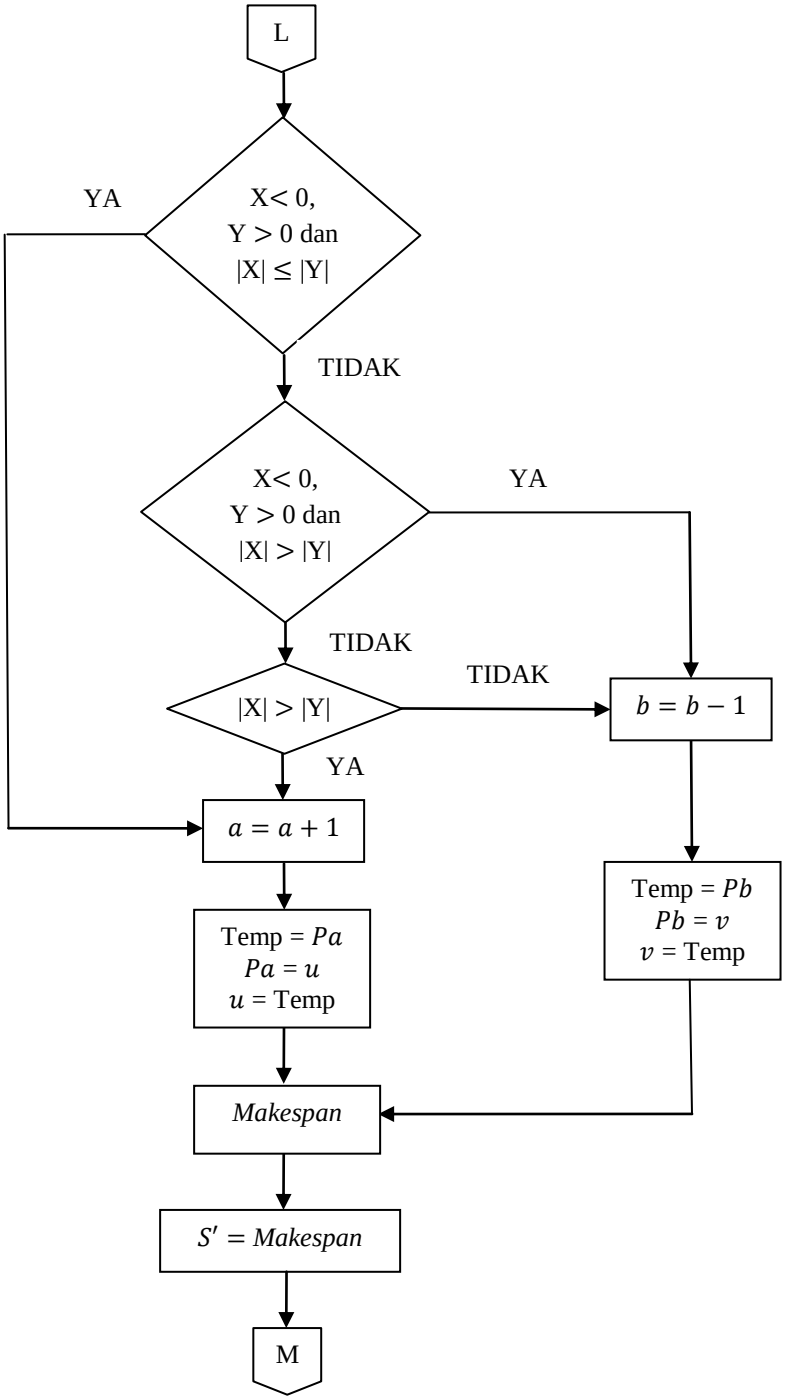
Gambar 4.6 menampilkan tahapan perhitungan untuk algoritma *Ho-Chang (HC)*, dengan menambahkan kompleksitas waktu tahapan perhitungan solusi awal dalam bentuk *input Makespan.*

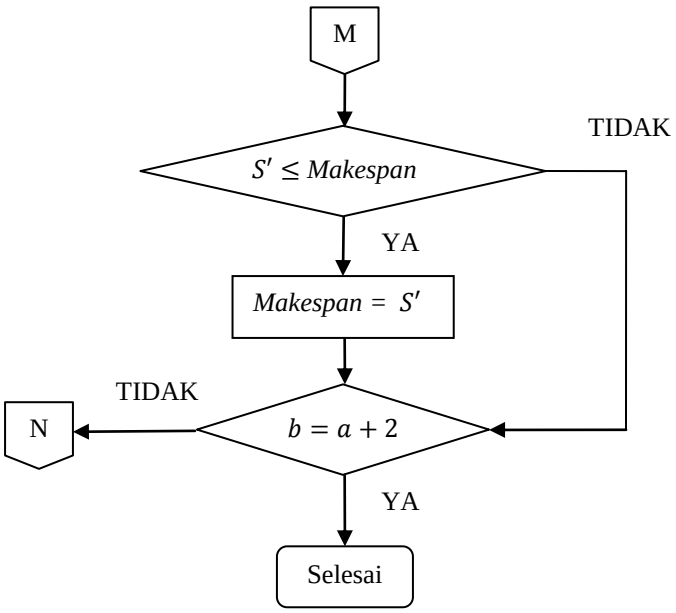

Mulai
Makespan
$k = 1$
$i = 1$
F
$j = 1$
$D_{ij}^{k} = t_{i(k+1)} - t_{jk}$
$j = j + 1$
TIDAK
$j > n$
YA
$i = i + 1$
TIDAK
$i > n$
YA
$fk = \left\{\left(\frac{1-0,1}{m-2}\right) \times (m-k-1)\right\} + 0,1$
E


E
$D_{ij}^{k} < 0$
TIDAK
$\delta_{ij} = 1$
YA
$\delta_{ij} = fk$
$k = k + 1$
$k > m - 1$
TIDAK
F
YA
$i = 1$
$j = 1$
I
$k = 1$
H
$d_{ij} \mathrel{+}= D_{ij}^{k} \,.\, \delta_{ij}$
$k = k + 1$
TIDAK
$k > m - 1$
YA
$j = j + 1$
G


G
$j > n$
TIDAK
H
YA
$i = i + 1$
$i > n$
TIDAK
I
YA
d_{ij}
$a = 1$
$b = n$
$l = a + 1$
N
$Pa = [t_{aj}]$
$Pl = [t_{lj}]$
K
d_{PaPl}
X = d_{PaPl}
J


J
X > d_{PaPl}
TIDAK
X = d_{PaPl}
K
YA
$l = b - 1$
TIDAK
$l = l + 1$
YA
$u = Pl$
$Pb = \lceil t_{bj} \rceil$
d_{PlPb}
Y = d_{PlPb}
TIDAK
Y < d_{PlPb}
Y = d_{PlPb}
YA
TIDAK
$l = b - 1$
YA
$v = Pl$
L


L
X< 0,
Y > 0 dan
|X| ≤ |Y|
YA
TIDAK
X< 0,
Y > 0 dan
|X| > |Y|
YA
TIDAK
|X| > |Y|
TIDAK
$b = b - 1$
YA
$a = a + 1$
Temp = Pa
$Pa = u$
u = Temp
Temp = Pb
$Pb = v$
v = Temp
Makespan
$S' = Makespan$
M



Gambar 4.6 *Flowchart* algoritma *Ho-Chang*

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan untuk algoritma *Ho-Chang (HC)*.

1. Kompleksitas untuk menentukan nilai D_{ij}^{k} :

- Untuk $k = 1$,

  Untuk $i = 1$,

  Untuk $j = 1,2, \ldots, n$; jumlah perhitungan = n kali

- Untuk $k = 1$,

  Untuk $i = 2$,

  Untuk $j = 1,2, \ldots, n$; jumlah perhitungan = n kali

…

- Untuk $k = 1$,

  Untuk $i = n$,

  Untuk $j = 1,2, \ldots, n$; jumlah perhitungan = n kali

- Untuk $k = 2$,

  Untuk $i = 1$,

  Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

…

- Untuk $k = m - 1$,

  Untuk $i = n$,

  Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

Jadi, jumlah perhitungannya → $T(n,m)_1 = (m-1) \times n^2 = (m-1)n^2$

2. Kompleksitas untuk menentukan nilai fk :

- Untuk $k = 1$,

  Untuk $fk = 1$,; jumlah perhitungan = 1 kali

- Untuk $k = 2$,

  Untuk $fk = 1$,; jumlah perhitungan = 1 kali

…

- Untuk $k = m - 1$,

  Untuk $fk = 1$,; jumlah perhitungan = 1 kali

Jadi, jumlah perhitungannya → $T(n,m)_2 = (m-1)$

3. Perhitungan untuk menentukan nilai δ_{ij}^{k} :

- Untuk $k = 1$,

  Untuk $i = 1$,

  Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

- Untuk $k = 1$,

  Untuk $i = 2$,

  Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

…

- Untuk $k = 1$,

  Untuk $i = n$,

Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

- Untuk $k = 2$,

  Untuk $i = 1$,

  Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

…

- Untuk $k = m - 1$,

  Untuk $i = n$,

  Untuk $j = 1,2, \dots, n$; jumlah perhitungan = n kali

Jadi, jumlah perhitungannya → $T(n,m)_3 = (m-1) \times n^2 = (m-1)n^2$

4. Kompleksitas untuk menentukan nilai d_{ij} :

- Untuk $i = 1$,

  Untuk $j = 1$,

  Untuk $k = 1,2, \dots, (m-1)$; jumlah perhitungan = $m - 1$ kali

- Untuk $i = 1$,

  Untuk $j = 2$,

  Untuk $k = 1,2, \dots, (m-1)$; jumlah perhitungan = $m - 1$ kali

…

- Untuk $i = 1$,

  Untuk $j = n$,

  Untuk $k = 1,2, \dots, (m-1)$; jumlah perhitungan = $m - 1$ kali

- Untuk $i = 2$,

  Untuk $j = 1$,

  Untuk $k = 1,2, \dots, (m-1)$; jumlah perhitungan = $m - 1$ kali

…

- Untuk $i = n$,

  Untuk $j = n$,

  Untuk $k = 1,2, \dots, (m-1)$; jumlah perhitungan = $m - 1$ kali

Jadi, jumlah perhitungannya → $T(n,m)_4 = (m-1) \times n^2 = (m-1)n^2$

5. Kompleksitas untuk menentukan nilai a dan b:

- Untuk $a = 1$,

  Untuk $l = 2, \dots, (n-1)$,

  $d_{12}, \dots, d_{1(n-1)}$; jumlah perhitungan = $n - 2$ kali

  Untuk $b = n$,

  Untuk $l = 2, \dots, (n-1)$,

  $d_{2n}, \dots, d_{(n-1)n}$; jumlah perhitungan = $n - 2$ kali

  Perhitungan syarat X, Y dan *makespan* jadwal baru terdapat 4 kompleksitas yang mungkin yakni:

  a. Ya + *makespan* = $1 + 1 + 3 + 2nm = 5 + 2nm$ kali
  b. Tidak → Ya + *makespan* = $1 + 1 + 1 + 3 + 2nm = 6 + 2nm$ kali
  c. Tidak → Tidak → Tidak + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali
  d. Tidak → Tidak → Ya + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali

  Dalam perhitungan untuk syarat X, Y dan *makespan* jadwal baru dipilih kompleksitas waktu yang paling lama yakni $7 + 2nm$ kali, hal ini dimaksudkan untuk menghitung kemungkinan waktu paling maksimum dari tahapan perhitungan tersebut. Sehingga, jumlah perhitungan yang dilakukan dalam proses diatas adalah

  $T(n, m) = 2(n-2) + (7 + 2nm)$

- Untuk $a = 2$,

  Untuk $l = 3, \dots, (n-1)$,

  $d_{23}, \dots, d_{2(n-1)}$; jumlah perhitungan = $n - 3$ kali

  Untuk $b = n$,

  Untuk $l = 3, \dots, (n-1)$,

  $d_{3n}, \dots, d_{(n-1)n}$; jumlah perhitungan = $n - 3$ kali

Perhitungan syarat X, Y dan *makespan* jadwal baru:

a. Ya + *makespan* = $1 + 1 + 3 + 2nm = 5 + 2nm$ kali
b. Tidak → Ya + *makespan* = $1 + 1 + 1 + 3 + 2nm = 6 + 2nm$ kali
c. Tidak → Tidak → Tidak + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali
d. Tidak → Tidak → Ya + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali

Sehingga, jumlah perhitungan yang dilakukan dalam proses diatas adalah

$T(n, m) = 2(n - 3) + (7 + 2nm)$

- Untuk $a = 3$,

  Untuk $l = 4, \dots, (n - 1)$,

  $d_{24}, \dots, d_{2(n-1)}$; jumlah perhitungan = $n - 4$ kali

Untuk $b = n$,

Untuk $l = 4, \dots, (n - 1)$,

$d_{4n}, \dots, d_{(n-1)n}$; jumlah perhitungan = $n - 4$ kali

Perhitungan syarat X, Y dan *makespan* jadwal baru:

a. Ya + *makespan* = $1 + 1 + 3 + 2nm = 5 + 2nm$ kali
b. Tidak → Ya + *makespan* = $1 + 1 + 1 + 3 + 2nm = 6 + 2nm$ kali
c. Tidak → Tidak → Tidak + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali
d. Tidak → Tidak → Ya + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali

Sehingga, jumlah perhitungan yang dilakukan dalam proses diatas adalah

$T(n, m) = 2(n - 4) + (7 + 2nm)$

…

- Untuk $a = n - 2$,

  Untuk $l = n - 1$,

  $d_{(n-2)(n-1)}$; jumlah perhitungan = 1 kali

Untuk $b = n$,

Untuk $l = n - 1$,

$d_{(n-1)n}$; jumlah perhitungan = 1 kali

Perhitungan syarat X, Y dan *makespan* jadwal baru:

a. Ya + *makespan* = $1 + 1 + 3 + 2nm = 5 + 2nm$ kali
b. Tidak → Ya + *makespan* = $1 + 1 + 1 + 3 + 2nm = 6 + 2nm$ kali
c. Tidak → Tidak → Tidak + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali
d. Tidak → Tidak → Ya + *makespan* = $1 + 1 + 1 + 1 + 3 + 2nm = 7 + 2nm$ kali

Sehingga, jumlah perhitungan yang dilakukan dalam proses diatas adalah

$T(n, m) = 2(1) + (7 + 2nm) = 9 + 2nm$

Dari perhitungan kompleksitas diatas membentuk deret aritmatika dengan $a = 2(n - 2) + (7 + 2nm)$ dan $b = -2$

$S_{(n,m)} = \frac{n}{2} + (2a + (n - 1)b) = 2n^2m + n^2 + 2n$

Jadi, jumlah perhitungannya → $T(n, m)_5 = 2n^2m + n^2 + 4n$

6. Kompleksitas untuk mendefinisikan *makespan* dari jadwal baru:

   $T(n, m)_6 = 1 + 1 + 1 = 3$

   Jumlah operasi penugasan terdapat tiga buah dan masing-masing dilakukan sebanyak satu kali.

Sehingga total kompleksitas waktu dari tahapan perhitungan algoritma *Ho-Chang* dengan solusi awal diperoleh

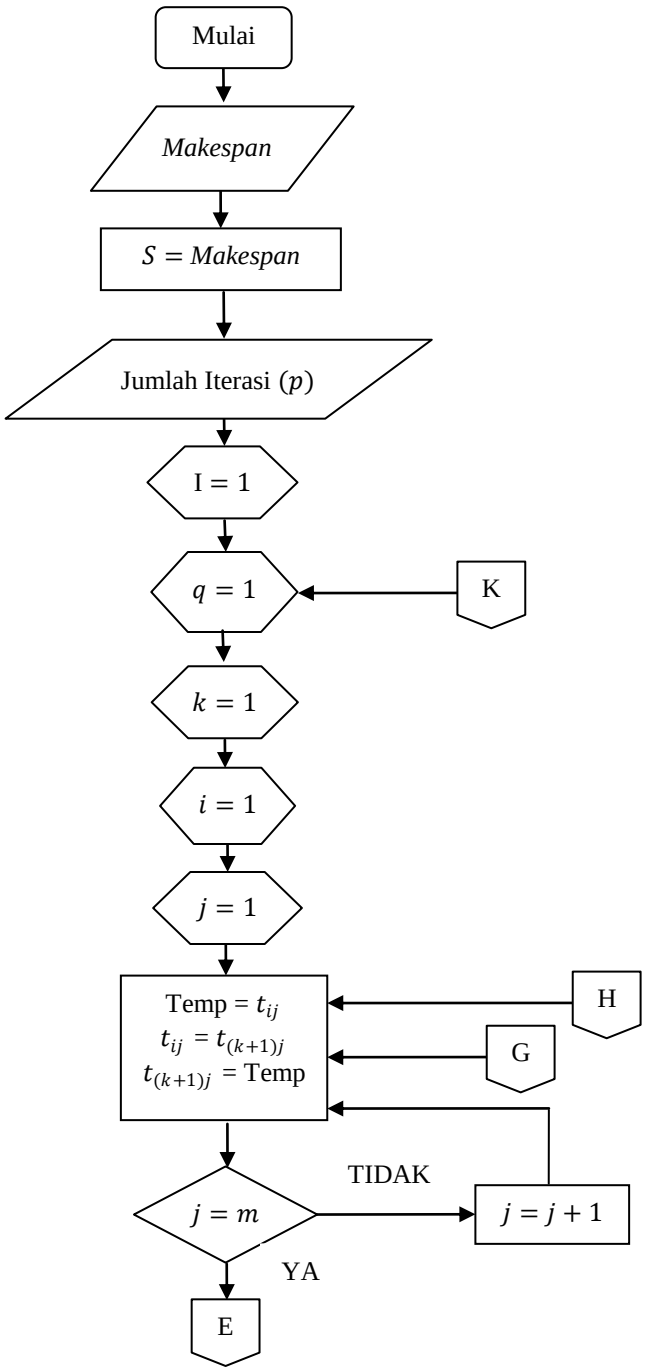
$$T(n, m) = 2nm + T(n, m)_1 + T(n, m)_2 + T(n, m)_3 + T(n, m)_4 + T(n, m)_5 + T(n, m)_6$$
$$= 2nm + (m - 1)n^2 + (m - 1) + (m - 1)n^2 + (m - 1)n^2 + (2n^2m + n^2 + 4n) + 3$$
$$= 5n^2m - 2n^2 + 2nm + 4n + m + 2$$

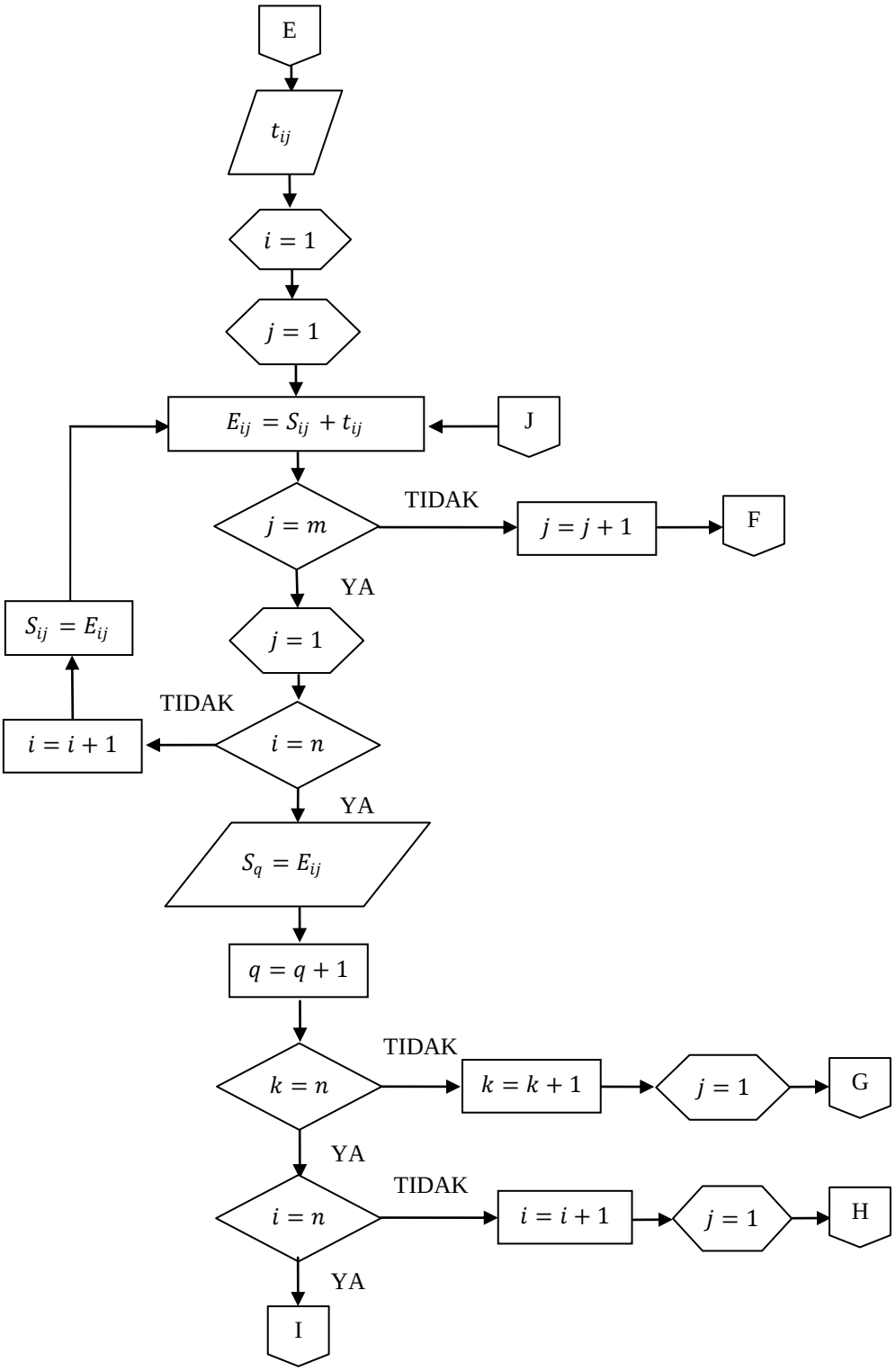
Karena

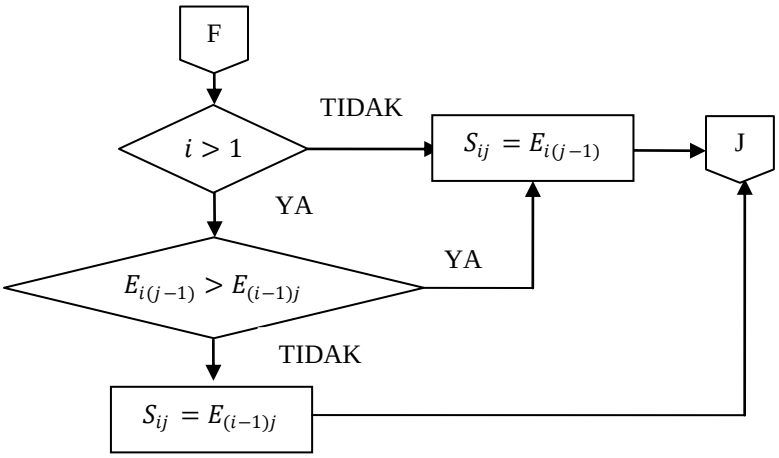
$5n^2m - 2n^2 + 2nm + 4n + m + 2 \leq 5n^2m + 2n^2m + 2n^2m + 4n^2m + n^2m + 2n^2m = 16n^2m$

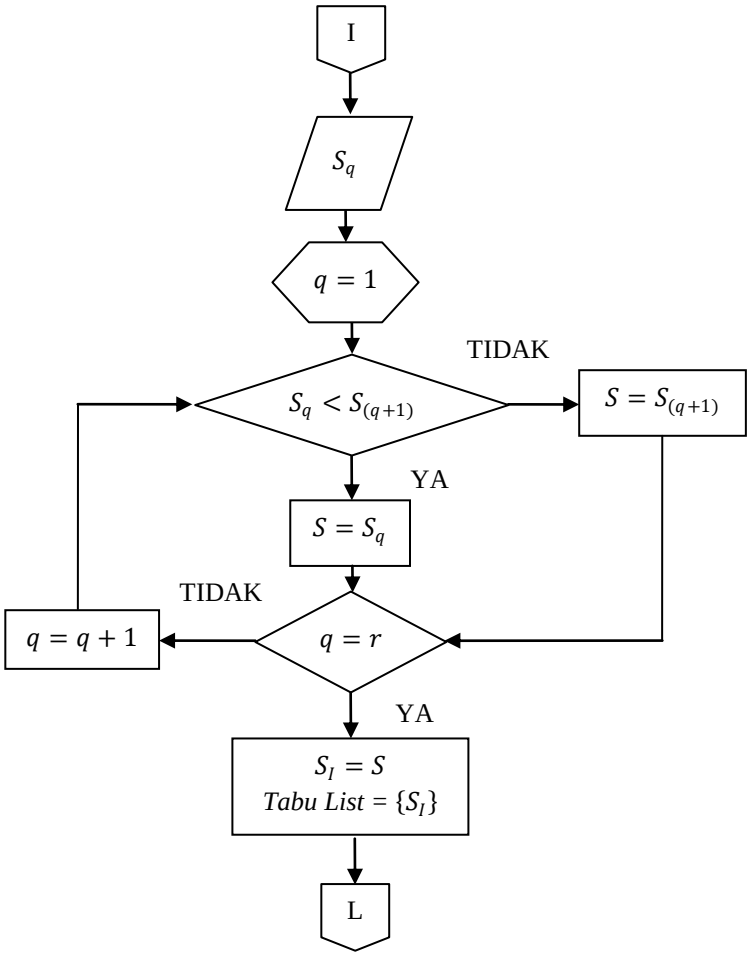
maka $O(n^2m)$

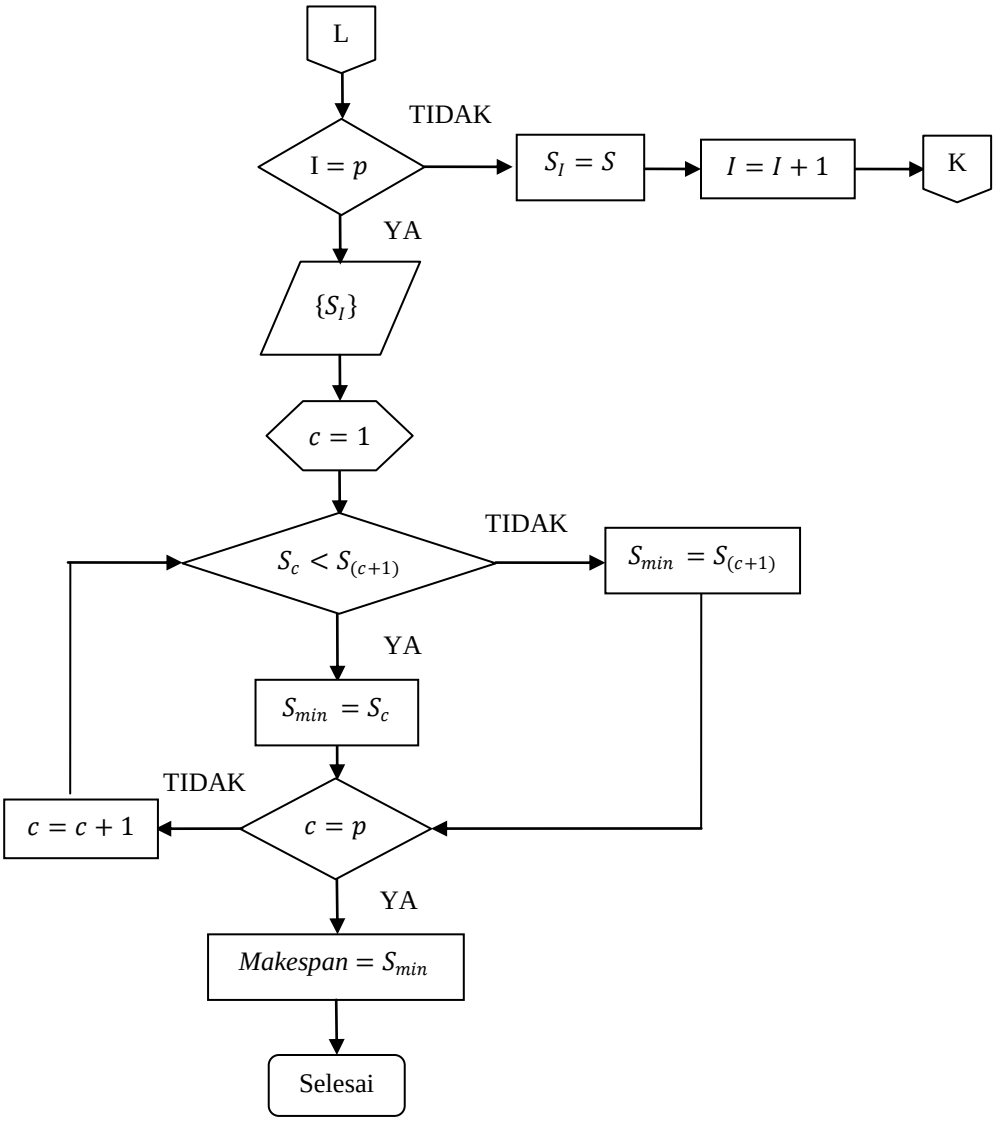
2. Algoritma *Tabu Search*




E
t_{ij}
$i = 1$
$j = 1$
$E_{ij} = S_{ij} + t_{ij}$
J
$j = m$
TIDAK
$j = j + 1$
F
YA
$j = 1$
$i = n$
TIDAK
$i = i + 1$
$S_{ij} = E_{ij}$
YA
$S_q = E_{ij}$
$q = q + 1$
$k = n$
TIDAK
$k = k + 1$
$j = 1$
G
YA
$i = n$
TIDAK
$i = i + 1$
$j = 1$
H
YA
I


F
TIDAK
$i > 1$
$S_{ij} = E_{i(j-1)}$
J
YA
YA
$E_{i(j-1)} > E_{(i-1)j}$
TIDAK
$S_{ij} = E_{(i-1)j}$


I
S_q
$q = 1$
TIDAK
$S_q < S_{(q+1)}$
$S = S_{(q+1)}$
YA
$S = S_q$
TIDAK
$q = q + 1$
$q = r$
YA
$S_I = S$
Tabu List = $\{S_I\}$
L



Gambar 4.7 *Flowchart* algoritma *Tabu Search*

Berikut perhitungan kompleksitas waktu dari tahapan perhitungan untuk algoritma *Tabu Search*.

1. Kompleksitas waktu dalam perhitungan tiap iterasi.

   Untuk $I = 1$,

   a. Perhitungan untuk tiap solusi yang terbentuk dari *neighborhood move*:

   - Untuk $i = 1$,

     Untuk $k = 1$,

Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali

- Untuk $i = 1$,
  Untuk $k = 2$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  …
- Untuk $i = 1$,
  Untuk $k = n$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  …
- Untuk $i = n$,
  Untuk $k = n$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  Jadi, jumlah perhitungannya → $T(n, m, r, p) = n^2 \times 3m = 3n^2m$

- Tiap solusi yang terbentuk dari *neighborhood move* dihitung nilai *makespan*nya dengan cara yang sama dengan pencarian *makespan* pada solusi awal maka kompleksitas waktunya sama yakni $T(n, m, r, p) = 2nm$.
  Sehingga jumlah perhitungannya → $T(n, m, r, p)_1 = (3n^2m) \times (2nm) = 6n^3m^2$

b. Perhitungan untuk evaluasi *makespan* yang optimal dari solusi yang dihasilkan dalam *neighborhood move*:

- Untuk $q = 1$; jumlah perhitungan = 4 kali
- Untuk $q = 2$; jumlah perhitungan = 4 kali
  …
- Untuk $q = r$; jumlah perhitungan = 4 kali
  Jadi, jumlah perhitungannya → $T(n, m, r, p)_2 = 4 \times r = 4r$

Sehingga total kompleksitas waktu dalam iterasi pertama yakni:

$$T(n, m, r, p) = T(n, m, r, p)_1 + T(n, m, r, p)_2$$
$$= 6n^3m^2 + 4r$$

Untuk $I = 2$,

a. Perhitungan untuk tiap solusi yang terbentuk dari *neighborhood move*:

- Untuk $i = 1$,
  Untuk $k = 1$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
- Untuk $i = 1$,
  Untuk $k = 2$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  …
- Untuk $i = 1$,
  Untuk $k = n$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  …
- Untuk $i = n$,
  Untuk $k = n$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  Jadi, jumlah perhitungannya $\rightarrow T(n, m, r, p) = n^2 \times 3m = 3n^2m$
- Tiap solusi yang terbentuk dari *neighborhood move* dihitung nilai *makespan*nya dengan cara yang sama dengan pencarian *makespan* pada solusi awal maka kompleksitas waktunya sama yakni $T(n, m, r, p) = 2nm$.
  Sehingga jumlah perhitungannya $\rightarrow T(n, m, r, p)_1 = (3n^2m) \times (2nm) = 6n^3m^2$

b. Perhitungan untuk evaluasi *makespan* yang optimal dari solusi yang dihasilkan dalam *neighborhood move*:

- Untuk $q = 1$; jumlah perhitungan = 4 kali
- Untuk $q = 2$; jumlah perhitungan = 4 kali
  …
- Untuk $q = r$; jumlah perhitungan = 4 kali
  Jadi, jumlah perhitungannya $\rightarrow T(n, m, r, p)_2 = 4 \times r = 4r$

Sehingga total kompleksitas waktu dalam iterasi kedua yakni:

$$T(n,m,r,p) = T(n,m,r,p)_1 + T(n,m,r,p)_2$$
$$= 6n^3m^2 + 4r$$

...

Untuk $I = p$,

a. Perhitungan untuk tiap solusi yang terbentuk dari *neighborhood move*:

- Untuk $i = 1$,
  Untuk $k = 1$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
- Untuk $i = 1$,
  Untuk $k = 2$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  …
- Untuk $i = 1$,
  Untuk $k = n$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  …
- Untuk $i = n$,
  Untuk $k = n$,
  Untuk $j = 1,2, \dots, m$; jumlah perhitungan = $3m$ kali
  Jadi, jumlah perhitungannya $\rightarrow T(n,m,r,p) = n^2 \times 3m = 3n^2m$
- Tiap solusi yang terbentuk dari *neighborhood move* dihitung nilai *makespan*nya dengan cara yang sama dengan pencarian *makespan* pada solusi awal maka kompleksitas waktunya sama yakni $T(n,m,r,p) = 2nm$.
  Sehingga jumlah perhitungannya $\rightarrow T(n,m,r,p)_1 = (3n^2m) \times (2nm) = 6n^3m^2$

b. Perhitungan untuk evaluasi *makespan* yang optimal dari solusi yang dihasilkan dalam *neighborhood move*:

- Untuk $q = 1$; jumlah perhitungan = 4 kali

- Untuk $q = 2$; jumlah perhitungan = 4 kali

  …

- Untuk $q = r$; jumlah perhitungan = 4 kali

  Jadi, jumlah perhitungannya → $T(n, m, r, p)_2 = 4 \times r = 4r$

Sehingga total kompleksitas waktu dalam iterasi kedua yakni:

$$T(n, m, r, p) = T(n, m, r, p)_1 + T(n, m, r, p)_2$$
$$= 6n^3m^2 + 4r$$

Jadi, total kompleksitas waktu sampai iterasi ke-p yakni:

$$T(n, m, r, p)_a = (6n^3m^2 + 4r) \times p$$
$$= (6n^3m^2 + 4r)p$$
$$= 6n^3m^2p + 4rp$$

2. Kompleksitas waktu untuk evaluasi *makespan* dalam *tabu list*:

- Untuk $c = 1$; jumlah perhitungannya$= 3$ kali
- Untuk $c = 2$; jumlah perhitungannya$= 3$ kali

  …

- Untuk $c = p$; jumlah perhitungannya$= 3$ kali

Sehingga total kompleksitas waktu untuk evaluasi *makespan* dalam *tabu list* yakni:

$$T(n, m, r, p)_b = 3p$$

Jadi, kompleksitas waktu untuk perhitungan menggunakan algoritma *Tabu Search* ialah

$$T(n, m, r, p) = T(n, m, r, p)_a + T(n, m, r, p)_b$$
$$= 6n^3m^2p + 4rp + 3p$$

Karena dalam skripsi kali ini banyaknya iterasi telah dibatasi oleh penulis yakni sebanyak 6 iterasi $(p = 6)$ dan nilai $r = C_2^n$,

$$r = C_2^n \rightarrow r = \frac{n!}{(n-2)!2!}$$
$$= \frac{(n-2)!\,(n-1)n}{(n-2)!\,2!}$$
$$= \frac{n^2 - n}{2}$$

Maka nilai kompleksitas waktu dari algoritma *Tabu Search* dapat dituliskan kembali sebagai berikut:

$$T(n,m) = 36n^3m^2 + 24\left(\frac{n^2-n}{2}\right) + 18$$

$$= 36n^3m^2 + 12n^2 - 12n + 18$$

Karena

$$36m^2 + 12n^2 - 12n + 18 \leq 36n^3m^2 + 12n^3m^2 + 12n^3m^2 + 18n^3m^2 = 78n^3m^2$$

maka $O(n^3m^2)$

4.2 Pembahasan

Dalam skripsi ini, algoritma *Ho-Chang* dan *Tabu Search* digunakan untuk mengoptimalkan waktu produksi jamu instan di industri Sari Hutani dengan bantuan aplikasi yang memanfaatkan bahasa pemrograman PHP. Kedua algoritma tersebut memerlukan sebuah solusi awal dalam melakukan perhitungannya, yang berbeda ialah algoritma *Ho-Chang* cenderung menggunakan persamaan dalam perhitungannya, sedangkan algoritma *Tabu search* menggunakan teknik peluang solusi dari ketetanggaan. Dalam hal ini penulis menggunakan jadwal yang pernah dipakai pada industri Sari Hutani sebagai solusi awal, karena industri tersebut hingga saat ini belum memiliki jadwal produksi yang tetap atau paten, sehingga sering kali menambah waktu operasional untuk memenuhi permintaan konsumen. Adapun alasan penulis menggunakan program PHP yakni karena PHP merupakan bahasa *open source* yang dapat digunakan di berbagai sistem operasi (*Linux, Unix, Macintosh,* dan *Windows*), selain itu juga tersedia banyak referensi pendukungnya.

Pada subbab 4.1 telah dijelaskan dengan perhitungan manual dari masing-masing algoritma. Algoritma *Ho-Chang* membutuhkan 6 iterasi dalam perhitungannya, karenanya penulis juga memberikan perlakuan yang sama terhadap algoritma *Tabu Search* yakni membatasi perhitungan sebanyak 6 iterasi. Hal ini bertujuan untuk menguji seberapa baik solusi yang dihasilkan oleh masing-masing

algoritma jika dikerjakan dalam jumlah iterasi yang sama. Dari perhitungan manual diperoleh jadwal yang meminimumkan *makespan* ialah sebagai berikut:

Tabel 4.7 Solusi Optimal dari Perhitungan Manual

No.	Algoritma	Jadwal	*Makespan*
1	*Ho-Chang*	$J_1 - J_2 - J_3 - J_5 - J_7 - J_8 - J_6 - J_4 - J_9$	950
2	*Tabu Search*	$J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	**925**

Menurut Tabel 4.7 dapat diketahui bahwa menurut hasil perhitungan manual atau penjadwalan dengan menggunakan algoritma *Tabu Search* menghasilkan nilai *makespan* yang lebih kecil 2,63% jika dibandingkan dengan nilai *makespan* yang dihasilkan oleh algoritma *Ho-Chang*. Artinya, penggunaan algoritma *Tabu Search* lebih efektif jika diterapkan pada penjadwalan produksi jamu instan di industri Sari Hutani, karena dapat mengurangi waktu operasional mesin dalam proses produksi sehingga dapat pula mengurangi biaya produksi.

Berbeda halnya jika ditinjau dari perhitungan kompleksitas waktu yang dihasilkan. Algoritma *Tabu Search* memiliki kompleksitas waktu asimptotik yakni $O(n^3m^2)$, maka termasuk dalam kelompok algoritma kubik. Sedangkan algoritma *Ho-Chang* dengan $O(n^2m)$, termasuk dalam kelompok algoritma kuadratik. Artinya, proses perhitungan menggunakan algoritma *Tabu Search* membutuhkan waktu yang lebih lama jika dibandingkan dengan proses perhitungan menggunakan algoritma *Ho-Chang*. Dengan kata lain menurut kompleksitas waktu yang diperoleh dapat dikatakan algoritma *Ho-Chang* lebih efisien dibandingkan dengan algoritma *Tabu Search* untuk diterapkan pada penjadwalan produksi jamu instan Sari Hutani.

Berdasarkan analisa diatas dapat disimpulkan, bahwa dalam penjadwalan produksi jamu instan di industri Sari Hutani, algoritma *Ho-Chang* merupakan algoritma yang efisien karena membutuhkan waktu yang lebih cepat dalam perhitungannya, namun tidak efektif karena jadwal yang dihasilkan kurang optimal. Sedangkan algoritma *Tabu Search* merupakan algoritma yang efektif karena

menghasilkan jadwal yang lebih optimal, namun tidak efisien karena membutuhkan waktu yang lebih lama dalam perhitungannya.

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dan pembahasan dapat disimpulkan bahwa:

1. Pada penjadwalan produksi jamu instan di industri Sari Hutani, nilai *makespan* minimum yang dihasilkan oleh algoritma *Ho-Chang* ialah 950 menit, diperoleh dengan urutan pekerjaan *(job)* salah satu diantaranya yaitu jamu instan pelancar asi → jamu instan kunci sirih → jamu instan temulawak → jamu instan kunyit asam → jamu instan som java → jamu instan diabetes → jamu instan sari urat → jamu instan sari jahe → jamu instan kolesterol. Sedangkan algoritma *Tabu Search* menghasilkan *makespan* 925 menit, diperoleh dengan urutan pekerjaan *(job)* salah satu diantaranya yaitu jamu instan kunci sirih → jamu instan pelancar asi → jamu instan temulawak → jamu instan sari jahe → jamu instan kunyit asam → jamu instan sari urat → jamu instan som java → jamu instan diabetes → jamu instan kolesterol.
2. Algoritma *Tabu Search* merupakan algoritma yang lebih efektif untuk diterapkan pada penjadwalan produksi jamu instan Sari Hutani dengan nilai *makespan* 925 menit. Namun lebih tidak efisien karena membutuhkan waktu komputasi yang lama dengan $O(n^3m^2)$.
3. Algoritma *Ho-Chang* merupakan algoritma yang lebih efisien untuk diterapkan pada penjadwalan produksi jamu instan Sari Hutani dengan kompleksitas waktu asimptotik $O(n^2m)$. Namun lebih tidak efektif karena menghasilkan nilai *makespan* yang kurang optimal.

5.2 Saran

Algoritma *Ho-Chang* dan *Tabu Search* merupakan algoritma yang dapat digunakan untuk menyelesaikan permasalahan optimasi dan masih terbuka bagi peneliti lain untuk mengaplikasikan kedua algoritma tersebut kedalam permasalahan optimasi lainnya selain *flowshop*, seperti *jobshop* dan *travelling salesman problem* (TSP). Penggunaan bahasa pemrograman selain PHP sebagai aplikasi juga masih terbuka bagi peneliti lainnya, seperti *Java Script* dan *Virtual Basic*.

DAFTAR PUSTAKA

Aditya, Alan Nur. 2011. *Jago PHP & MySQL*. Penerbit Dunia Komputer: Bekasi.

Baker, Kenneth.1974. *Introduction to Sequencing and Scheduling*. Jhon Wiley and Sons,Inc.

Berlianty, I. & Arifin, M. 2010. *Teknik-teknik Optimasi Heuristik.* Penerbit Graha Ilmu: Yogyakarta.

Bondal, A. A. 2008. *Artificial Immune Systems Applied to Job Shop Scheduling.* Fakultas Mesin dan Teknologi College Russ, Universitas Ohio:Ohio.

Conway, R. W., Maxwell, W. L., & Miller, L. W. 1967. *Theory of Scheduling*. Addison-Wesley, Reading, MA.

Ginting, R. 2009. *Penjadwalan Mesin*. Penerbit Graha Ilmu:Yogyakarta.

Ginting, R. & S.,Ginting, H. 2006. *Study Aplikasi Metode Artifical Immune System dalam Penjadwalan Flow Shop*. Departemen Teknik Mesin Fakultas Teknik Universitas Sumatera Utara:Medan.

Glover, F. 1998. Tabu Search-Wellsprings and Challenges. *European Journal of Operational Research*. University of Colorado: USA.

Ho, Johnny C., Chang, Yih-Long. 1991. A New Heuristic for the n-job, M-machine Flow Shop Problem. *European Journal of Operational Research.* North Holland.

Masruroh, Nisa. Tanpa tahun. *Analisa Penjadwalan Produksi dengan Menggunakan Metode Campbell Dudeck Smith, Palmer dan Danennbring di PT.Loka Refraktoris Surabaya*. Teknik Industri FTI UPN: Jatim.

Panggabean, H. P. 2005. *Penjadwalan Jobshop Statik dengan Algoritma Tabu Search.* Jurusan Ilmu Komputer FMIPA Universitas Katolik Parahyangan: Bandung.

Ravetti, M.G., Nakamura, F.G., Meneses, C.N., Resende, M.G., Mateus, & G.R.Pardalos,P., 2006. Hybrid Heuristics for the Permutation Flow Shop Problem. *AT&T Labs Research Technical Report* TD-6V9MEV,Shannon Laboratory, Florham Park,NJ 07932 USA.[serial online]

http:/citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.69.7239&rep=repl&type=pdf. [12 Desember 2011]

Shofwatul'uyun. 2009. *Kompleksitas Algoritma*. Teknik Informatika UIN Kalijaga. [serial online] http://www.scribd.com/doc/38673595/null. [20 November 2012]

Soetanto, Tessa V. & Soetanto, Danny P. 1999. Penjadualan Flowshop Dengan Algoritma Genetika. *Jurnal Teknik Industri FT UK Petra*.

Wibowo, Ferry Wahyu. 2011. *Matematika Diskrit: Kompleksitas Algoritma.* STMIK Amikom yogyakarta.

A. HASIL PERHITUNGAN *OVERALL REVISED GAPS* ALGORITMA *HO-CHANG*

- $d_{11} = (10 \times 1) + (35 \times 1) + (-50 \times 0{,}4) + (55 \times 1) = 80$
- $d_{12} = (10 \times 1) + (40 \times 1) + (-30 \times 0{,}4) + (55 \times 1) = 93$
- $d_{13} = (10 \times 1) + (35 \times 1) + (-35 \times 0{,}4) + (55 \times 1) = 86$
- $d_{14} = (10 \times 1) + (20 \times 1) + (-55 \times 0{,}4) + (55 \times 1) = 63$
- $d_{15} = (10 \times 1) + (35 \times 1) + (-45 \times 0{,}4) + (55 \times 1) = 82$
- $d_{16} = (10 \times 1) + (20 \times 1) + (-45 \times 0{,}4) + (55 \times 1) = 67$
- $d_{17} = (10 \times 1) + (25 \times 1) + (-30 \times 0{,}4) + (55 \times 1) = 78$
- $d_{18} = (10 \times 1) + (25 \times 1) + (-35 \times 0{,}4) + (55 \times 1) = 76$
- $d_{19} = (10 \times 1) + (35 \times 1) + (-55 \times 0{,}4) + (55 \times 1) = 78$
- $d_{21} = (5 \times 1) + (15 \times 1) + (-50 \times 0{,}4) + (75 \times 1) = 75$
- $d_{22} = (5 \times 1) + (20 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 88$
- $d_{23} = (5 \times 1) + (15 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 81$
- $d_{24} = (5 \times 1) + (0 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 58$
- $d_{25} = (5 \times 1) + (15 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 77$
- $d_{26} = (5 \times 1) + (0 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 62$
- $d_{27} = (5 \times 1) + (5 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 73$
- $d_{28} = (5 \times 1) + (5 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 71$
- $d_{29} = (5 \times 1) + (15 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 73$
- $d_{31} = (10 \times 1) + (20 \times 1) + (-50 \times 0{,}4) + (75 \times 1) = 85$
- $d_{32} = (10 \times 1) + (25 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 98$
- $d_{33} = (10 \times 1) + (20 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 91$
- $d_{34} = (10 \times 1) + (5 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 68$
- $d_{35} = (10 \times 1) + (20 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 87$
- $d_{36} = (10 \times 1) + (5 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 72$
- $d_{37} = (10 \times 1) + (10 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 83$
- $d_{38} = (10 \times 1) + (10 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 81$
- $d_{39} = (10 \times 1) + (20 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 83$

- $d_{41} = (25 \times 1) + (40 \times 1) + (-50 \times 0{,}4) + (75 \times 1) = 120$
- $d_{42} = (25 \times 1) + (45 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 133$
- $d_{43} = (25 \times 1) + (40 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 126$
- $d_{44} = (25 \times 1) + (25 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 103$
- $d_{45} = (25 \times 1) + (40 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 122$
- $d_{46} = (25 \times 1) + (25 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 107$
- $d_{47} = (25 \times 1) + (30 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 118$
- $d_{48} = (25 \times 1) + (30 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 116$
- $d_{49} = (25 \times 1) + (40 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 118$
- $d_{51} = (10 \times 1) + (30 \times 1) + (-50 \times 0{,}4) + (75 \times 1) = 95$
- $d_{52} = (10 \times 1) + (35 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 108$
- $d_{53} = (10 \times 1) + (30 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 101$
- $d_{54} = (10 \times 1) + (15 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 78$
- $d_{55} = (10 \times 1) + (30 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 97$
- $d_{56} = (10 \times 1) + (15 \times 1) + (-45 \times 0{,}4) + (75 \times 1) = 82$
- $d_{57} = (10 \times 1) + (20 \times 1) + (-30 \times 0{,}4) + (75 \times 1) = 93$
- $d_{58} = (10 \times 1) + (20 \times 1) + (-35 \times 0{,}4) + (75 \times 1) = 91$
- $d_{59} = (10 \times 1) + (30 \times 1) + (-55 \times 0{,}4) + (75 \times 1) = 93$
- $d_{61} = (25 \times 1) + (30 \times 1) + (-50 \times 0{,}4) + (85 \times 1) = 120$
- $d_{62} = (25 \times 1) + (35 \times 1) + (-30 \times 0{,}4) + (85 \times 1) = 133$
- $d_{63} = (25 \times 1) + (30 \times 1) + (-35 \times 0{,}4) + (85 \times 1) = 126$
- $d_{64} = (25 \times 1) + (15 \times 1) + (-55 \times 0{,}4) + (85 \times 1) = 103$
- $d_{65} = (25 \times 1) + (30 \times 1) + (-45 \times 0{,}4) + (85 \times 1) = 122$
- $d_{66} = (25 \times 1) + (15 \times 1) + (-45 \times 0{,}4) + (85 \times 1) = 107$
- $d_{67} = (25 \times 1) + (20 \times 1) + (-30 \times 0{,}4) + (85 \times 1) = 118$
- $d_{68} = (25 \times 1) + (20 \times 1) + (-35 \times 0{,}4) + (85 \times 1) = 116$
- $d_{69} = (25 \times 1) + (30 \times 1) + (-55 \times 0{,}4) + (85 \times 1) = 118$
- $d_{71} = (20 \times 1) + (15 \times 1) + (-50 \times 0{,}4) + (85 \times 1) = 100$
- $d_{72} = (20 \times 1) + (20 \times 1) + (-30 \times 0{,}4) + (85 \times 1) = 113$
- $d_{73} = (20 \times 1) + (15 \times 1) + (-35 \times 0{,}4) + (85 \times 1) = 106$

- $d_{74} = (20 \times 1) + (0 \times 1) + (-55 \times 0{,}4) + (85 \times 1) = 83$
- $d_{75} = (20 \times 1) + (15 \times 1) + (-45 \times 0{,}4) + (85 \times 1) = 102$
- $d_{76} = (20 \times 1) + (0 \times 1) + (-45 \times 0{,}4) + (85 \times 1) = 87$
- $d_{77} = (20 \times 1) + (5 \times 1) + (-30 \times 0{,}4) + (85 \times 1) = 98$
- $d_{78} = (20 \times 1) + (5 \times 1) + (-35 \times 0{,}4) + (85 \times 1) = 96$
- $d_{79} = (20 \times 1) + (15 \times 1) + (-55 \times 0{,}4) + (85 \times 1) = 98$
- $d_{81} = (20 \times 1) + (20 \times 1) + (-50 \times 0{,}4) + (55 \times 1) = 75$
- $d_{82} = (20 \times 1) + (25 \times 1) + (-30 \times 0{,}4) + (55 \times 1) = 88$
- $d_{83} = (20 \times 1) + (20 \times 1) + (-35 \times 0{,}4) + (55 \times 1) = 81$
- $d_{84} = (20 \times 1) + (5 \times 1) + (-55 \times 0{,}4) + (55 \times 1) = 58$
- $d_{85} = (20 \times 1) + (20 \times 1) + (-45 \times 0{,}4) + (55 \times 1) = 77$
- $d_{86} = (20 \times 1) + (5 \times 1) + (-45 \times 0{,}4) + (55 \times 1) = 62$
- $d_{87} = (20 \times 1) + (10 \times 1) + (-30 \times 0{,}4) + (55 \times 1) = 73$
- $d_{88} = (20 \times 1) + (10 \times 1) + (-35 \times 0{,}4) + (55 \times 1) = 71$
- $d_{89} = (20 \times 1) + (20 \times 1) + (-55 \times 0{,}4) + (55 \times 1) = 73$
- $d_{91} = (10 \times 1) + (40 \times 1) + (-50 \times 0{,}4) + (105 \times 1) = 135$
- $d_{92} = (10 \times 1) + (45 \times 1) + (-30 \times 0{,}4) + (105 \times 1) = 148$
- $d_{93} = (10 \times 1) + (40 \times 1) + (-35 \times 0{,}4) + (105 \times 1) = 141$
- $d_{94} = (10 \times 1) + (25 \times 1) + (-55 \times 0{,}4) + (105 \times 1) = 118$
- $d_{95} = (10 \times 1) + (40 \times 1) + (-45 \times 0{,}4) + (105 \times 1) = 137$
- $d_{96} = (10 \times 1) + (25 \times 1) + (-45 \times 0{,}4) + (105 \times 1) = 122$
- $d_{97} = (10 \times 1) + (30 \times 1) + (-30 \times 0{,}4) + (105 \times 1) = 133$
- $d_{98} = (10 \times 1) + (30 \times 1) + (-35 \times 0{,}4) + (105 \times 1) = 131$
- $d_{99} = (10 \times 1) + (40 \times 1) + (-55 \times 0{,}4) + (105 \times 1) = 133$

B. HASIL PERTUKARAN *JOB* UNTUK TIAP ITERASI PADA ALGORITMA *TABU SEARCH*

Tabel A. *Neighborhood Search* pada Iterasi Pertama

Pertukaran	Urutan *job*	*makespan*
***job* 1 dengan *job* 2**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	**925**
***job*1 dengan *job* 3**	$J_3 - J_2 - J_1 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	935
***job* 1 dengan *job* 4**	$J_4 - J_2 - J_3 - J_1 - J_5 - J_6 - J_7 - J_8 - J_9$	970
***job* 1 dengan *job* 5**	$J_5 - J_2 - J_3 - J_4 - J_1 - J_6 - J_7 - J_8 - J_9$	945
***job* 1 dengan *job* 6**	$J_6 - J_2 - J_3 - J_4 - J_5 - J_1 - J_7 - J_8 - J_9$	960
***job* 1 dengan *job* 7**	$J_7 - J_2 - J_3 - J_4 - J_5 - J_6 - J_1 - J_8 - J_9$	940
***job* 1 dengan *job* 8**	$J_8 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_1 - J_9$	945
***job* 1 dengan *job* 9**	$J_9 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_1$	955
***job* 2 dengan *job* 3**	$J_1 - J_3 - J_2 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	950
***job* 2 dengan *job* 4**	$J_1 - J_4 - J_3 - J_2 - J_5 - J_6 - J_7 - J_8 - J_9$	950
***job* 2 dengan *job* 5**	$J_1 - J_5 - J_3 - J_4 - J_2 - J_6 - J_7 - J_8 - J_9$	950
***job* 2 dengan *job* 6**	$J_1 - J_6 - J_3 - J_4 - J_5 - J_2 - J_7 - J_8 - J_9$	950
***job* 2 dengan *job* 7**	$J_1 - J_7 - J_3 - J_4 - J_5 - J_6 - J_2 - J_8 - J_9$	950
***job* 2 dengan *job* 8**	$J_1 - J_8 - J_3 - J_4 - J_5 - J_6 - J_7 - J_2 - J_9$	950
***job* 2 dengan *job* 9**	$J_1 - J_9 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_2$	950
***job* 3 dengan *job* 4**	$J_1 - J_2 - J_4 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	950
***job* 3 dengan *job* 5**	$J_1 - J_2 - J_5 - J_4 - J_3 - J_6 - J_7 - J_8 - J_9$	950
***job* 3 dengan *job* 6**	$J_1 - J_2 - J_6 - J_4 - J_5 - J_3 - J_7 - J_8 - J_9$	950
***job* 3 dengan *job* 7**	$J_1 - J_2 - J_7 - J_4 - J_5 - J_6 - J_3 - J_8 - J_9$	950
***job* 3 dengan *job* 8**	$J_1 - J_2 - J_8 - J_4 - J_5 - J_6 - J_7 - J_3 - J_9$	950
***job* 3 dengan *job* 9**	$J_1 - J_2 - J_9 - J_4 - J_5 - J_6 - J_7 - J_8 - J_3$	950
***job* 4 dengan *job* 5**	$J_1 - J_2 - J_3 - J_5 - J_4 - J_6 - J_7 - J_8 - J_9$	950
***job* 4 dengan *job* 6**	$J_1 - J_2 - J_3 - J_6 - J_5 - J_4 - J_7 - J_8 - J_9$	950
***job* 4 dengan *job* 7**	$J_1 - J_2 - J_3 - J_7 - J_5 - J_6 - J_4 - J_8 - J_9$	950

***job* 4 dengan *job* 8**	$J_1 - J_2 - J_3 - J_8 - J_5 - J_6 - J_7 - J_4 - J_9$	950
***job* 4 dengan *job* 9**	$J_1 - J_2 - J_3 - J_9 - J_5 - J_6 - J_7 - J_8 - J_4$	950
***job* 5 dengan *job* 6**	$J_1 - J_2 - J_3 - J_4 - J_6 - J_5 - J_7 - J_8 - J_9$	950
***job* 5 dengan *job* 7**	$J_1 - J_2 - J_3 - J_4 - J_7 - J_6 - J_5 - J_8 - J_9$	950
***job* 5 dengan *job* 8**	$J_1 - J_2 - J_3 - J_4 - J_8 - J_6 - J_7 - J_5 - J_9$	950
***job* 5 dengan *job* 9**	$J_1 - J_2 - J_3 - J_4 - J_9 - J_6 - J_7 - J_8 - J_5$	950
***job* 6 dengan *job* 7**	$J_1 - J_2 - J_3 - J_4 - J_5 - J_7 - J_6 - J_8 - J_9$	950
***job* 6 dengan *job* 8**	$J_1 - J_2 - J_3 - J_4 - J_5 - J_9 - J_7 - J_8 - J_6$	950
***job* 6 dengan *job* 9**	$J_1 - J_2 - J_3 - J_4 - J_5 - J_9 - J_7 - J_8 - J_6$	950
***job* 7 dengan *job* 8**	$J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_8 - J_7 - J_9$	950
***job* 7 dengan *job* 9**	$J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_9 - J_8 - J_7$	950
***job* 8 dengan *job* 9**	$J_1 - J_2 - J_3 - J_4 - J_5 - J_6 - J_7 - J_9 - J_8$	950

Tabel B. *Neighborhood Search* pada Iterasi Kedua

Pertukaran	**Urutan *job***	***makespan***
***job* 2 dengan *job* 3**	$J_3 - J_1 - J_2 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	935
***job* 2 dengan *job* 4**	$J_4 - J_1 - J_3 - J_2 - J_5 - J_6 - J_7 - J_8 - J_9$	970
***job* 2 dengan *job* 5**	$J_5 - J_1 - J_3 - J_4 - J_2 - J_6 - J_7 - J_8 - J_9$	945
***job* 2 dengan *job* 6**	$J_6 - J_1 - J_3 - J_4 - J_5 - J_2 - J_7 - J_8 - J_9$	960
***job* 2 dengan *job* 7**	$J_7 - J_1 - J_3 - J_4 - J_5 - J_6 - J_2 - J_8 - J_9$	940
***job* 2 dengan *job* 8**	$J_8 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_2 - J_9$	945
***job* 2 dengan *job* 9**	$J_9 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_2$	955
***job* 1 dengan *job* 3**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	**925**
***job* 1 dengan *job* 4**	$J_2 - J_4 - J_3 - J_1 - J_5 - J_6 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 5**	$J_2 - J_5 - J_3 - J_4 - J_1 - J_6 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 6**	$J_2 - J_6 - J_3 - J_4 - J_5 - J_1 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 7**	$J_2 - J_7 - J_3 - J_4 - J_5 - J_6 - J_1 - J_8 - J_9$	955
***job* 1 dengan *job* 8**	$J_2 - J_8 - J_3 - J_4 - J_5 - J_6 - J_7 - J_1 - J_9$	925

***job* 1 dengan *job* 9**	$J_2 - J_9 - J_3 - J_4 - J_5 - J_6 - J_7 - J_8 - J_1$	925
***job* 3 dengan *job* 4**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	930
***job* 3 dengan *job* 5**	$J_2 - J_1 - J_5 - J_4 - J_3 - J_6 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 6**	$J_2 - J_1 - J_6 - J_4 - J_5 - J_3 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 7**	$J_2 - J_1 - J_7 - J_4 - J_5 - J_6 - J_3 - J_8 - J_9$	925
***job* 3 dengan *job* 8**	$J_2 - J_1 - J_8 - J_4 - J_5 - J_6 - J_7 - J_3 - J_9$	925
***job* 3 dengan *job* 9**	$J_2 - J_1 - J_9 - J_4 - J_5 - J_6 - J_7 - J_8 - J_3$	925
***job* 4 dengan *job* 5**	$J_2 - J_1 - J_3 - J_5 - J_4 - J_6 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 6**	$J_2 - J_1 - J_3 - J_6 - J_5 - J_4 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 7**	$J_2 - J_1 - J_3 - J_7 - J_5 - J_6 - J_4 - J_8 - J_9$	925
***job* 4 dengan *job* 8**	$J_2 - J_1 - J_3 - J_8 - J_5 - J_6 - J_7 - J_4 - J_9$	925
***job* 4 dengan *job* 9**	$J_2 - J_1 - J_3 - J_9 - J_5 - J_6 - J_7 - J_8 - J_4$	925
***job* 5 dengan *job* 6**	$J_2 - J_1 - J_3 - J_4 - J_6 - J_5 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 7**	$J_2 - J_1 - J_3 - J_4 - J_7 - J_6 - J_5 - J_8 - J_9$	925
***job* 5 dengan *job* 8**	$J_2 - J_1 - J_3 - J_4 - J_8 - J_6 - J_7 - J_5 - J_9$	925
***job* 5 dengan *job* 9**	$J_2 - J_1 - J_3 - J_4 - J_9 - J_6 - J_7 - J_8 - J_5$	925
***job* 6 dengan *job* 7**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_7 - J_6 - J_8 - J_9$	925
***job* 6 dengan *job* 8**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_8 - J_7 - J_6 - J_9$	925
***job* 6 dengan *job* 9**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_9 - J_7 - J_8 - J_6$	925
***job* 7 dengan *job* 8**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_8 - J_7 - J_9$	925
***job* 7 dengan *job* 9**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_9 - J_8 - J_7$	925
***job* 8 dengan *job* 9**	$J_2 - J_1 - J_3 - J_4 - J_5 - J_6 - J_7 - J_9 - J_8$	925

Tabel C. *Neighborhood Search* pada Iterasi Ketiga

Pertukaran	**Urutan *job***	***makespan***
***job* 2 dengan *job* 3**	$J_3 - J_2 - J_1 - J_4 - J_5 - J_6 - J_7 - J_8 - J_9$	935
***job* 2 dengan *job* 4**	$J_4 - J_3 - J_1 - J_2 - J_5 - J_6 - J_7 - J_8 - J_9$	970
***job* 2 dengan *job* 5**	$J_5 - J_3 - J_1 - J_4 - J_2 - J_6 - J_7 - J_8 - J_9$	945

***job* 2 dengan *job* 6**	$J_6 - J_3 - J_1 - J_4 - J_5 - J_2 - J_7 - J_8 - J_9$	960
***job* 2 dengan *job* 7**	$J_7 - J_3 - J_1 - J_4 - J_5 - J_6 - J_2 - J_8 - J_9$	940
***job* 2 dengan *job* 8**	$J_8 - J_3 - J_1 - J_4 - J_5 - J_6 - J_7 - J_2 - J_9$	945
***job* 2 dengan *job* 9**	$J_9 - J_3 - J_1 - J_4 - J_5 - J_6 - J_7 - J_8 - J_2$	955
***job* 3 dengan *job* 4**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	**925**
***job* 3 dengan *job* 5**	$J_2 - J_5 - J_1 - J_4 - J_3 - J_6 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 6**	$J_2 - J_6 - J_1 - J_4 - J_5 - J_3 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 7**	$J_2 - J_7 - J_1 - J_4 - J_5 - J_6 - J_3 - J_8 - J_9$	925
***job* 3 dengan *job* 8**	$J_2 - J_8 - J_1 - J_4 - J_5 - J_6 - J_7 - J_3 - J_9$	925
***job* 3 dengan *job* 9**	$J_2 - J_9 - J_1 - J_4 - J_5 - J_6 - J_7 - J_8 - J_3$	955
***job* 1 dengan *job* 4**	$J_2 - J_3 - J_4 - J_1 - J_5 - J_6 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 5**	$J_2 - J_3 - J_5 - J_4 - J_1 - J_6 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 6**	$J_2 - J_3 - J_6 - J_4 - J_5 - J_1 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 7**	$J_2 - J_3 - J_7 - J_4 - J_5 - J_6 - J_1 - J_8 - J_9$	925
***job* 1 dengan *job* 8**	$J_2 - J_3 - J_8 - J_4 - J_5 - J_6 - J_7 - J_1 - J_9$	925
***job* 1 dengan *job* 9**	$J_2 - J_3 - J_9 - J_4 - J_5 - J_6 - J_7 - J_8 - J_1$	925
***job* 4 dengan *job* 5**	$J_2 - J_3 - J_1 - J_5 - J_4 - J_6 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 6**	$J_2 - J_3 - J_1 - J_6 - J_5 - J_4 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 7**	$J_2 - J_3 - J_1 - J_7 - J_5 - J_6 - J_4 - J_8 - J_9$	925
***job* 4 dengan *job* 8**	$J_2 - J_3 - J_1 - J_8 - J_5 - J_6 - J_7 - J_4 - J_9$	925
***job* 4 dengan *job* 9**	$J_2 - J_3 - J_1 - J_9 - J_5 - J_6 - J_7 - J_8 - J_4$	925
***job* 5 dengan *job* 6**	$J_2 - J_3 - J_1 - J_4 - J_6 - J_5 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 7**	$J_2 - J_3 - J_1 - J_4 - J_7 - J_6 - J_5 - J_8 - J_9$	925
***job* 5 dengan *job* 8**	$J_2 - J_3 - J_1 - J_4 - J_8 - J_6 - J_7 - J_5 - J_9$	925
***job* 5 dengan *job* 9**	$J_2 - J_3 - J_1 - J_4 - J_9 - J_6 - J_7 - J_8 - J_5$	925
***job* 6 dengan *job* 7**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_7 - J_6 - J_8 - J_9$	925
***job* 6 dengan *job* 8**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_8 - J_7 - J_6 - J_9$	925
***job* 6 dengan *job* 9**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_9 - J_7 - J_8 - J_6$	925

***job* 7 dengan *job* 8**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_6 - J_8 - J_7 - J_9$	925
***job* 7 dengan *job* 9**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_6 - J_9 - J_8 - J_7$	925
***job* 8 dengan *job* 9**	$J_2 - J_3 - J_1 - J_4 - J_5 - J_6 - J_7 - J_9 - J_8$	925

Tabel D. *Neighborhood Search* pada Iterasi Keempat

Pertukaran	**Urutan *job***	***makespan***
***job* 2 dengan *job* 4**	$J_4 - J_2 - J_1 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	970
***job* 2 dengan *job* 3**	$J_3 - J_4 - J_1 - J_2 - J_5 - J_6 - J_7 - J_8 - J_9$	935
***job* 2 dengan *job* 5**	$J_5 - J_4 - J_1 - J_3 - J_2 - J_6 - J_7 - J_8 - J_9$	945
***job* 2 dengan *job* 6**	$J_6 - J_4 - J_1 - J_3 - J_5 - J_2 - J_7 - J_8 - J_9$	960
***job* 2 dengan *job* 7**	$J_7 - J_4 - J_1 - J_3 - J_5 - J_6 - J_2 - J_8 - J_9$	945
***job* 2 dengan *job* 8**	$J_8 - J_4 - J_1 - J_3 - J_5 - J_6 - J_7 - J_2 - J_9$	945
***job* 2 dengan *job* 9**	$J_9 - J_4 - J_1 - J_3 - J_5 - J_6 - J_7 - J_8 - J_2$	955
***job* 4 dengan *job* 1**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	**925**
***job* 4 dengan *job* 5**	$J_2 - J_5 - J_1 - J_3 - J_4 - J_6 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 6**	$J_2 - J_6 - J_1 - J_3 - J_5 - J_4 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 7**	$J_2 - J_7 - J_1 - J_3 - J_5 - J_6 - J_4 - J_8 - J_9$	925
***job* 4 dengan *job* 8**	$J_2 - J_8 - J_1 - J_3 - J_5 - J_6 - J_7 - J_4 - J_9$	925
***job* 4 dengan *job* 9**	$J_2 - J_9 - J_1 - J_3 - J_5 - J_6 - J_7 - J_8 - J_4$	955
***job* 1 dengan *job* 5**	$J_2 - J_4 - J_5 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 6**	$J_2 - J_4 - J_6 - J_3 - J_5 - J_1 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 7**	$J_2 - J_4 - J_7 - J_3 - J_5 - J_6 - J_1 - J_8 - J_9$	925
***job* 1 dengan *job* 8**	$J_2 - J_4 - J_8 - J_3 - J_5 - J_6 - J_7 - J_1 - J_9$	925
***job* 1 dengan *job* 9**	$J_2 - J_4 - J_9 - J_3 - J_5 - J_6 - J_7 - J_8 - J_1$	925
***job* 3 dengan *job* 5**	$J_2 - J_4 - J_1 - J_5 - J_3 - J_6 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 6**	$J_2 - J_4 - J_1 - J_6 - J_5 - J_3 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 7**	$J_2 - J_4 - J_1 - J_7 - J_5 - J_6 - J_3 - J_8 - J_9$	925
***job* 3 dengan *job* 8**	$J_2 - J_4 - J_1 - J_8 - J_5 - J_6 - J_7 - J_3 - J_9$	925

***job* 3 dengan *job* 9**	$J_2 - J_4 - J_1 - J_9 - J_5 - J_6 - J_7 - J_8 - J_3$	925
***job* 5 dengan *job* 6**	$J_2 - J_4 - J_1 - J_3 - J_6 - J_5 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 7**	$J_2 - J_4 - J_1 - J_3 - J_7 - J_6 - J_5 - J_8 - J_9$	925
***job* 5 dengan *job* 8**	$J_2 - J_4 - J_1 - J_3 - J_8 - J_6 - J_7 - J_5 - J_9$	925
***job* 5 dengan *job* 9**	$J_2 - J_4 - J_1 - J_3 - J_9 - J_6 - J_7 - J_8 - J_5$	925
***job* 6 dengan *job* 7**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_7 - J_6 - J_8 - J_9$	925
***job* 6 dengan *job* 8**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_8 - J_7 - J_6 - J_9$	925
***job* 6 dengan *job* 9**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_9 - J_7 - J_8 - J_6$	925
***job* 7 dengan *job* 8**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_6 - J_8 - J_7 - J_9$	925
***job* 7 dengan *job* 9**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_6 - J_9 - J_8 - J_7$	925
***job* 8 dengan *job* 9**	$J_2 - J_4 - J_1 - J_3 - J_5 - J_6 - J_7 - J_9 - J_8$	925

Tabel E. *Neighborhood Search* pada Iterasi Kelima

Pertukaran	**Urutan *job***	***makespan***
***job* 2 dengan *job* 4**	$J_4 - J_1 - J_2 - J_3 - J_5 - J_6 - J_7 - J_8 - J_9$	970
***job* 2 dengan *job* 3**	$J_3 - J_1 - J_4 - J_2 - J_5 - J_6 - J_7 - J_8 - J_9$	935
***job* 2 dengan *job* 5**	$J_5 - J_1 - J_4 - J_3 - J_2 - J_6 - J_7 - J_8 - J_9$	945
***job* 2 dengan *job* 6**	$J_6 - J_1 - J_4 - J_3 - J_5 - J_2 - J_7 - J_8 - J_9$	960
***job* 2 dengan *job* 7**	$J_7 - J_1 - J_4 - J_3 - J_5 - J_6 - J_2 - J_8 - J_9$	940
***job* 2 dengan *job* 8**	$J_8 - J_1 - J_4 - J_3 - J_5 - J_6 - J_7 - J_2 - J_9$	945
***job* 2 dengan *job* 9**	$J_9 - J_1 - J_4 - J_3 - J_5 - J_6 - J_7 - J_8 - J_2$	955
***job* 1 dengan *job* 5**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	**925**
***job* 1 dengan *job* 6**	$J_2 - J_6 - J_4 - J_3 - J_5 - J_1 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 7**	$J_2 - J_7 - J_4 - J_3 - J_5 - J_6 - J_1 - J_8 - J_9$	925
***job* 1 dengan *job* 8**	$J_2 - J_8 - J_4 - J_3 - J_5 - J_6 - J_7 - J_1 - J_9$	925
***job* 1 dengan *job* 9**	$J_2 - J_9 - J_4 - J_3 - J_5 - J_6 - J_7 - J_8 - J_1$	925
***job* 4 dengan *job* 5**	$J_2 - J_1 - J_5 - J_3 - J_4 - J_6 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 6**	$J_2 - J_1 - J_6 - J_3 - J_5 - J_4 - J_7 - J_8 - J_9$	925

***job* 4 dengan *job* 7**	$J_2 - J_1 - J_7 - J_3 - J_5 - J_6 - J_4 - J_8 - J_9$	925
***job* 4 dengan *job* 8**	$J_2 - J_1 - J_8 - J_3 - J_5 - J_6 - J_7 - J_4 - J_9$	925
***job* 4 dengan *job* 9**	$J_2 - J_1 - J_9 - J_3 - J_5 - J_6 - J_7 - J_8 - J_4$	925
***job* 3 dengan *job* 5**	$J_2 - J_1 - J_4 - J_5 - J_3 - J_6 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 6**	$J_2 - J_1 - J_4 - J_6 - J_5 - J_3 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 7**	$J_2 - J_1 - J_4 - J_7 - J_5 - J_6 - J_3 - J_8 - J_9$	925
***job* 3 dengan *job* 8**	$J_2 - J_1 - J_4 - J_8 - J_5 - J_6 - J_7 - J_3 - J_9$	925
***job* 3 dengan *job* 9**	$J_2 - J_1 - J_4 - J_9 - J_5 - J_6 - J_7 - J_8 - J_3$	925
***job* 5 dengan *job* 6**	$J_2 - J_1 - J_4 - J_3 - J_6 - J_5 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 7**	$J_2 - J_1 - J_4 - J_3 - J_7 - J_6 - J_5 - J_8 - J_9$	925
***job* 5 dengan *job* 8**	$J_2 - J_1 - J_4 - J_3 - J_8 - J_6 - J_7 - J_5 - J_9$	925
***job* 5 dengan *job* 9**	$J_2 - J_1 - J_4 - J_3 - J_9 - J_6 - J_7 - J_8 - J_5$	925
***job* 6 dengan *job* 7**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_7 - J_6 - J_8 - J_9$	925
***job* 6 dengan *job* 8**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_8 - J_7 - J_6 - J_9$	925
***job* 6 dengan *job* 9**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_9 - J_7 - J_8 - J_6$	925
***job* 7 dengan *job* 8**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_6 - J_8 - J_7 - J_9$	925
***job* 7 dengan *job* 9**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_6 - J_9 - J_8 - J_7$	925
***job* 8 dengan *job* 9**	$J_2 - J_1 - J_4 - J_3 - J_5 - J_6 - J_7 - J_9 - J_8$	925

Tabel F. *Neighborhood Search* pada Iterasi Keenam

Pertukaran	**Urutan *job***	***makespan***
***job* 2 dengan *job* 5**	$J_5 - J_2 - J_4 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	945
***job* 2 dengan *job* 4**	$J_4 - J_5 - J_2 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	970
***job* 2 dengan *job* 3**	$J_3 - J_5 - J_4 - J_2 - J_1 - J_6 - J_7 - J_8 - J_9$	935
***job* 2 dengan *job* 6**	$J_6 - J_5 - J_4 - J_3 - J_1 - J_2 - J_7 - J_8 - J_9$	960
***job* 2 dengan *job* 7**	$J_7 - J_5 - J_4 - J_3 - J_1 - J_6 - J_2 - J_8 - J_9$	940
***job* 2 dengan *job* 8**	$J_8 - J_5 - J_4 - J_3 - J_1 - J_6 - J_7 - J_2 - J_9$	945
***job* 2 dengan *job* 9**	$J_9 - J_5 - J_4 - J_3 - J_1 - J_6 - J_7 - J_8 - J_2$	955

***job* 5 dengan *job* 4**	$J_2 - J_4 - J_5 - J_3 - J_1 - J_6 - J_7 - J_8 - J_9$	**925**
***job* 5 dengan *job* 3**	$J_2 - J_3 - J_4 - J_5 - J_1 - J_6 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 6**	$J_2 - J_6 - J_4 - J_3 - J_1 - J_5 - J_7 - J_8 - J_9$	925
***job* 5 dengan *job* 7**	$J_2 - J_7 - J_4 - J_3 - J_1 - J_6 - J_5 - J_8 - J_9$	925
***job* 5 dengan *job* 8**	$J_2 - J_8 - J_4 - J_3 - J_1 - J_6 - J_7 - J_5 - J_9$	925
***job* 5 dengan *job* 9**	$J_2 - J_9 - J_4 - J_3 - J_1 - J_6 - J_7 - J_8 - J_5$	925
***job* 4 dengan *job* 6**	$J_2 - J_5 - J_6 - J_3 - J_1 - J_4 - J_7 - J_8 - J_9$	925
***job* 4 dengan *job* 7**	$J_2 - J_5 - J_7 - J_3 - J_1 - J_6 - J_4 - J_8 - J_9$	925
***job* 4 dengan *job* 8**	$J_2 - J_5 - J_8 - J_3 - J_1 - J_6 - J_7 - J_4 - J_9$	925
***job* 4 dengan *job* 9**	$J_2 - J_5 - J_9 - J_3 - J_1 - J_6 - J_7 - J_8 - J_4$	925
***job* 3 dengan *job* 6**	$J_2 - J_5 - J_4 - J_6 - J_1 - J_3 - J_7 - J_8 - J_9$	925
***job* 3 dengan *job* 7**	$J_2 - J_5 - J_4 - J_7 - J_1 - J_6 - J_3 - J_8 - J_9$	925
***job* 3 dengan *job* 8**	$J_2 - J_5 - J_4 - J_8 - J_1 - J_6 - J_7 - J_3 - J_9$	925
***job* 3 dengan *job* 9**	$J_2 - J_5 - J_4 - J_9 - J_1 - J_6 - J_7 - J_8 - J_3$	925
***job* 1 dengan *job* 6**	$J_2 - J_5 - J_4 - J_3 - J_6 - J_1 - J_7 - J_8 - J_9$	925
***job* 1 dengan *job* 7**	$J_2 - J_5 - J_4 - J_3 - J_7 - J_6 - J_1 - J_8 - J_9$	925
***job* 1 dengan *job* 8**	$J_2 - J_5 - J_4 - J_3 - J_8 - J_6 - J_7 - J_1 - J_9$	925
***job* 1 dengan *job* 9**	$J_2 - J_5 - J_4 - J_3 - J_9 - J_6 - J_7 - J_8 - J_1$	925
***job* 6 dengan *job* 7**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_7 - J_6 - J_8 - J_9$	925
***job* 6 dengan *job* 8**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_8 - J_7 - J_6 - J_9$	925
***job* 6 dengan *job* 9**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_9 - J_7 - J_8 - J_6$	925
***job* 7 dengan *job* 8**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_6 - J_8 - J_7 - J_9$	925
***job* 7 dengan *job* 9**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_6 - J_9 - J_8 - J_7$	925
***job* 8 dengan *job* 9**	$J_2 - J_5 - J_4 - J_3 - J_1 - J_6 - J_7 - J_9 - J_8$	925