PENYELESAIAN MASALAH PENJADWALAN *FLOWSHOP* DENGAN ALGORITMA *ITERATED GREEDY* DAN ALGORITMA *SOCIAL COGNITIVE OPTIMIZATION*

SKRIPSI

Oleh
Irman Efendi
NIM 121810101001

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PENYELESAIAN MASALAH PENJADWALAN *FLOWSHOP* DENGAN ALGORITMA *ITERATED GREEDY* DAN ALGORITMA *SOCIAL COGNITIVE OPTIMIZATION*

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat
untuk menyelesaikan Program Studi Matematika (S1)
dan mencapai gelar Sarjana Sains

Oleh
Irman Efendi
NIM 121810101001

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2016

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ayah Hantono dan Ibu Sukarsih yang telah memberikan cinta dan kasih sayangnya.
2. Keluarga besarku yang selalu memberikan dukungan;
3. Guru-guru sejak taman kanak-kanak sampai perguruan tinggi yang telah membimbing dan memberikan ilmu yang bermanfaat;
4. Teman-teman Bathic’s 12 yang selalu memberikan bantuan.
5. Almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember, SMAN 1 Paiton, SMP Negeri 3 Kraksaan, MI Mambaul Ulum Kraksaan, dan TK Mambaul Ulum.

MOTTO

"Sesungguhnya Allah tidak akan mengubah nasib suatu kaum
Kecuali kaum itu sendiri yang mengubah
Apa yang pada diri mereka"
(terjemahan Surat Ar-Rad Ayat 11)[1]

[1] Departemen Agama Republik Indonesia. 2004. *Al-Qur'an* dan Terjemahannya. Bandung: CV Penerbit Diponegoro

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Irman Efendi

NIM : 121810101001

Menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Penyelesaian Masalah Penjadwalan *Flowshop* Dengan Algoritma *Iterated Greedy* dan Algoritma *Social Cognitive Optimization*" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi manapun 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, Desember 2016

Yang menyatakan,

Irman Efendi

NIM 121810101001

SKRIPSI

PENYELESAIAN MASALAH PENJADWALAN *FLOWSHOP* DENGAN ALGORITMA *ITERATED GREEDY* DAN ALGORITMA *SOCIAL COGNITIVE OPTIMIZATION*

Oleh

Irman Efendi

NIM 121810101001

Pembimbing:

Dosen Pembimbing Utama : Kusbudiono, S.Si., M.Si.

Dosen Pembimbing Anggota : Ahmad Kamsyakawuni, S.Si., M.Kom.

PENGESAHAN

Skripsi berjudul "Penyelesaian Masalah Penjadwalan *Flowshop* dengan Algoritma *Iterated Greedy* dan Algoritma *Social Cognitive Optimiation*" telah diuji dan disahkan pada:

hari, tanggal :

tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Tim Penguji:

Ketua,	Sekretaris,
Kusbudiono, S.Si., M.Si. NIP. 197704302005011001	Ahmad Kamsayakawuni, S.Si., M.Kom. NIP. 197211291998021001
Dosen Penguji I,	Dosen Penguji II,
Kosala Dwidja Purnomo, S.Si., M.Si. NIP. 196908281998021001	Dr. Mohamad Fatekurohman, S. Si., M.Si NIP. 196906061998031001

Mengesahkan

Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Jember

Drs. Sujito, Ph. D.
NIP. 196102041987111001

PRAKATA

Puji syukur kehadirat Tuhan Yang Maha Esa atas segala kuasa-Nya sehingga penulis dapat menyelesaiakan tugas akhir yang berjudul "Penyelesaian Masalah Penjadwalan *Flowshop* dengan Algoritma *Iterated Greedy* dan Algoritma *Social Cognitive Optimization*". Penulisan tugas akhir ini dilakukan guna memenuhi salah satu syarat untuk menyelesaikan Program Studi Matematika (S1) dan mencapai gelar sarjana Sains pada Jurusan Matematika FMIPA Universitas Jember.

Pada kesempatan ini, dengan segala hormat penulis mengucapkan terima kasih kepada:

1. Bapak Kusbudiono, S.Si., M.Si. selaku dosen pembimbing utama dan Bapak Ahmad Kamsyakawuni, S.Si., M.Kom.
2. Bapak Kosala Dwidja Purnomo, S.Si., M.Si. dan Dr. Mohamad Fatekurohman, S. Si., M.Si selaku dosen penguji.
3. Bapak Drs. Sujito, Ph. D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam.
4. Keluarga Bathic's 12 yang telah banyak membantu.
5. Dosen FMIPA dan guru-guruku mulai dari TK, MI, SMP dan SMA yang telah banyak memberikan ilmu kepada saya.

Penulis menyadari bahwa penulisan ini masih jauh dari sempurna. Oleh karena itu, penulis mengharap kritik dan saran demi kesempurnaan penelitian selanjutnya. Semoga tugas akhir ini dapat bermanfaat bagi kita semua.

Jember, Desember 2016 Penulis

RINGKASAN

Penyelesaian Masalah Penjadwalan *Flowshop* dengan Algoritma *Iterated Greedy* dan Algoritma *Social Cognitive Optimization*. Irman Efendi, 1218101001; 2016; 66 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penjadwalan merupakan pengalokasian sumber daya yang terbatas untuk sejumlah pekerjaan. Adapun tujuan dari penjadwalan produksi yaitu meminimasi *makespan*. Salah satu bentuk penjadwalan adalah penjadwalan *flowshop*. Penjadwalan *flowshop* adalah penentuan urutan *job* yang memiliki lintasan yang sama.

Penelitian ini menggunakan beberapa data yaitu data sekunder dan data simulasi. Data sekunder yang digunakan adalah data produksi sepatu UD. Duta Kulit Laila (2014). Penelitian ini menyelesaikan masalah penjadwalan *flowshop* dengan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization*. Selain untuk mengetahui hasil dari penerapan kedua algoritma tersebut dalam menyelesaikan masalah penjadwalan *flowshop*, penelitian ini juga bertujuan untuk membandingkan kedua algoritma berdasarkan hasil *makespan*, tingkat kecepatan kekonvergenan dan *running time*.

Penelitian ini dilakukan melalui beberapa langkah, pengambilan data kemudian melakukan penjadwalan menggunakan kedua algoritma. Langkah selanjutnya adalah membandingkan hasil dari kedua algoritma berdasarkan nilai *makespan*, tingkat kecepatan kekonvergenan dan *running time* sehingga dapat disimpulkan dari perbandingan tersebut.

Berdasarkan hasil penelitian, algoritma *Iterated Greedy* memiliki efisiensi yang lebih baik ditinjau dari nilai *makespan* yang diperoleh dari algoritma *Social Cognitive Optimization*. Hasil *makespan* optimal yang diperoleh dari masing-masing algoritma yaitu 1508 menit untuk data (8 *job*, 8 mesin), hal ini menunjukan kedua algoritma tersebut memiliki tingkat efektifitas yang sama, tetapi untuk data

simulasi algoritma *Iterated Greedy* menunjukan *nilai makespan* yang lebih baik daripada algoritma *Social Cognitive Optimization*. Berdasarkan dari tingkat kecepatan kekonvergenan algoritma *Iterated Greedy* memiliki efisiensi lebih cepat dalam menemukan suatu nilai dibandingkan dengan algoritma *Social Cognitive Optimization*.

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Pada hakikatnya setiap perusahaan mempunyai tujuan yang sama yaitu mendapat keuntungan yang maksimal dengan pengeluaran biaya produksi seminimum mungkin. Hal tersebut dapat dicapai dengan cara menyelesaikan suatu pekerjaan atau produk yang dihasilkan sesuai harapan konsumen dan diselesaikan tepat waktu. Strategi yang dapat digunakan oleh perusahaan tersebut yaitu dengan melakukan penjadwalan. Penjadwalan dapat menjadi kunci dalam strategi perusahaan tersebut.

Penjadwalan merupakan pengalokasian sumber daya yang terbatas untuk sejumlah pekerjaan. Permasalahan yang muncul pada penjadwalan terjadi apabila pada tahapan operasi tertentu beberapa atau seluruh pekerjaan membutuhkan stasiun kerja yang sama sehingga perlu adanya pengurutan pekerjaan dalam suatu produksi. Salah satu bentuk penjadwalan yaitu penjadwalan *flowshop,* penjadwalan *flowshop* merupakan proses penentuan urutan *job* yang memiliki lintasan produk yang sama. Penjadwalan produksi memiliki tujuan untuk memininasi *makespan* (total waktu yang dibutuhkan untuk menyelesaikan seluruh pekerjaan).

Pendekatan yang dapat digunakan untuk menyelesaikan permasalahan *flowshop* adalah dengan metode metaheuristik. Metaheuristik adalah metode untuk mencari solusi yang memadukan interaksi antara prosedur pencarian *local* dan strategi yang lebih tinggi untuk menciptakan proses yang mampu keluar dari titik-titik *local optimum* dan melakukan pencarian di ruang solusi untuk menemukan solusi global. Salah satu optimasi metaheuristik yaitu *Iterated Greedy*. *Iterated Greedy* merupakan algoritma metaheuristik dalam proses penyelesaiannya terdapat tahap destruktif dan tahap konstruktif. Algoritma *Iterated Greedy* telah berhasil diterapkan pada permasalahan *flowshop* yang dilakukan oleh Ribas *et al*. (2011). Penelitian tersebut menyimpulkan bahwa algoritma *Iterated Greedy* berhasil

menemukan solusi baru yang lebih baik dari solusi terbaik yang diketahui. Selain itu, algoritma *Iterated Greedy* cukup sederhana dan sangat kompetitif.

Pendekatan berikutnya yang akan digunakan adalah algoritma *Social Cognitive Optimization* (SCO). Algoritma SCO adalah algoritma metaheuristik yang didasarkan pada teori *social cognitive* atau perilaku manusia. Algoritma SCO telah digunakan untuk menyelesaikan masalah *nonlinear programming* yang dilakukan oleh Xie *et al*., (2002). Pada penelitian tersebut, algoritma SCO dibandingkan dengan algoritma Genetika. Xie *et al*. memberikan kesimpulan bahwa algoritma SCO mampu menghasilkan solusi dengan kualitas yang baik dengan waktu proses yang lebih cepat.

Berdasarkan uraian tersebut, penulis tertarik untuk menerapkan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* pada penjadwalan *flowshop*. Kemudian kedua algoritma dibandingkan untuk mengetahui algoritma yang paling baik. Kriteria perbandingan didasarkan pada nilai *makespan*, *running time* dan tingkat kecepatan kekonvergenan. Data yang digunakan pada penelitian ini yaitu data sekunder dari Laila (2014) tentang produksi Sepatu UD. Selain data dari Laila (2014), peneliti juga menggunakan beberapa data simulasi.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, maka permasalahan yang dibahas adalah sebagai berikut:

a. Bagaimana penerapan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* pada penjadwalan *flowshop*?
b. Bagaimana perbandingan hasil kedua algoritma berdasarkan aspek *makespan*, *running time* dan tingkat kecepatan kekonvergenannya?

1.3 Batasan Masalah

Batasan-batasan masalah yang terdapat dalam tugas akhir ini sebagai berikut:

a. Setiap *job* memiliki *ready time* yang sama.
b. Waktu membawa *job* ke mesin diabaikan.

1.4 Tujuan

Adapun tujuan dari penelitian ini adalah:

a. Menerapkan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* pada penjadwalan *flowshop*.
b. Mengetahui perbandingan hasil dari kedua algoritma berdasarkan *makespan, running time* dan tingkat kecepatan kekonvergenan.

1.5 Manfaat

Manfaat dari penelitian ini adalah untuk menambah wawasan tentang algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* dalam menyelesaikan masalah *flowshop.* Selain itu, peneliti juga mendapatkan urutan *job* yang optimal dengan nilai *makespan* minimum dari kedua algoritma tersebut.

BAB 2. TINJAUAN PUSTAKA

Pada bab ini dibahas tentang teori-teori yang mendasari penelitian, yaitu mengenai Penjadwalan (*Scheduling)*, Penjadwalan *Flowshop* (*Flowshop Scheduling*), Algoritma *Iterated Greedy*, Algoritma *Social Cognitive Optimization* dan Kriteria Kekonvergenan.

2.1 Penjadwalan (*Scheduling*)

Scheduling (penjadwalan) merupakan masalah pengalokasian sumber daya yang terbatas untuk mengerjakan sejumlah pekerjaan (Ginting, 2009). Sedangkan menurut Fang (1994) penjadwalan (*Scheduling*) adalah masalah perencanaan waktu dan kombinasi untuk mendapatkan keluaran (*output*).

Tujuan penjadwalan secara umum Baker (1974) adalah:

a. Meningkatkan produktivitas mesin, yaitu dengan mengurangi waktu menganggur mesin.
b. Mengurangi terhadap persediaan barang setengah jadi, dengan mengurangi rata-rata pekerjaan yang menunggu dalam antrian karena mesin sibuk oleh pekerjaan lain.
c. Mengurangi keterlambatan (*tardiness*). Dalam banyak hal, beberapa atau semua pekerjaan mempunyai batas waktu penyelesaian (*duedate*). Apabila suatu pekerjaan melewati batas waktu tersebut, maka akan dikenai pinalti. Keterlambatan dapat diperkecil dengan mengurangi maksimal *tardiness* atau mengurangi pekerjaan yang terlambat (*number of tardy job*).

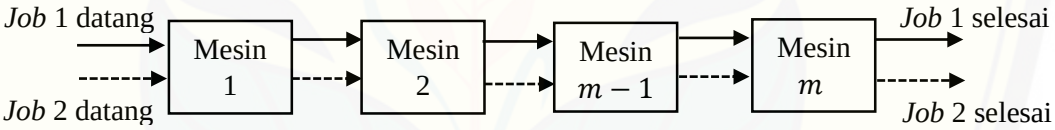
2.2 Penjadwalan *Flowshop* (*Flowshop Scheduling*)

Penjadwalan *flowshop* adalah proses penentuan urutan pekerjaan yang memiliki jalur produk yang sama. Pada pola *flowshop*, operasi dari suatu *job* hanya dapat bergerak satu arah, yaitu dari proses awal di mesin awal sampai proses akhir dimesin dan jumlah tahapan proses umumnya sama dengan jumlah jenis mesin yang digunakan. Penjadwalan *flowshop* ada karena terbatasnya

sumber daya yaitu mesin, beberapa pekerjaan yang harus diselesaikan, waktu dari pekerjaan yang bervariasi, dan adanya urutan pekerjaan yang harus dipenuhi. Umumnya penjadwalan *flowshop* terdiri dari beberapa mesin (m) dan mempunyai sejumlah *job* yang harus dikerjakan (n) serta waktu proses per-unit *job* pada mesin j, (dengan $t_{ij} = 1, \dots, n\,; j = 1, \dots, m$). Jadi permasalahan Penjadwalan merupakan fungsi untuk menentukan urutan *job* yang memberikan solusi terbaik berupa jadwal yang paling optimal. Terdapat target utama yang ingin dicapai melalui penjadwalan *flowshop* ini yaitu jumlah output yang dihasilkan (*throughput*) berupa *makespan*. Penjadwalan *flowshop* didefinisikan sebagai penjadwalan dimana setiap *job* mempunyai pola aliran atau rute proses yang tetap pada seluruh mesin.

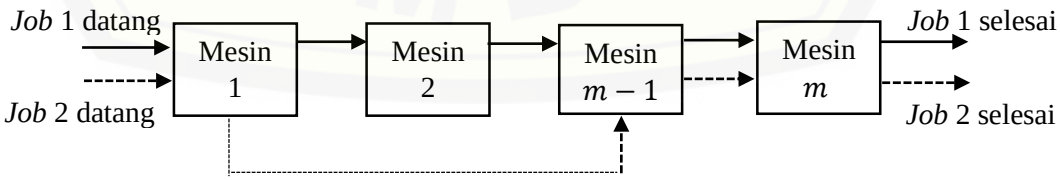
Menurut Baker dan Trietsch (2009), *flowshop* dibedakan menjadi dua yaitu:

a. *Pure flowshop,* berbagai *job* akan mengalir pada lini produksi yang sama atau *job* yang akan datang dikerjakan di semua mesin. Seperti, masing-masing *job* melalui mesin 1, setelah itu mesin 2, dan seterusnya sampai mesin m.



Gambar 2.1 Pola aliran *Pure Flowshop*

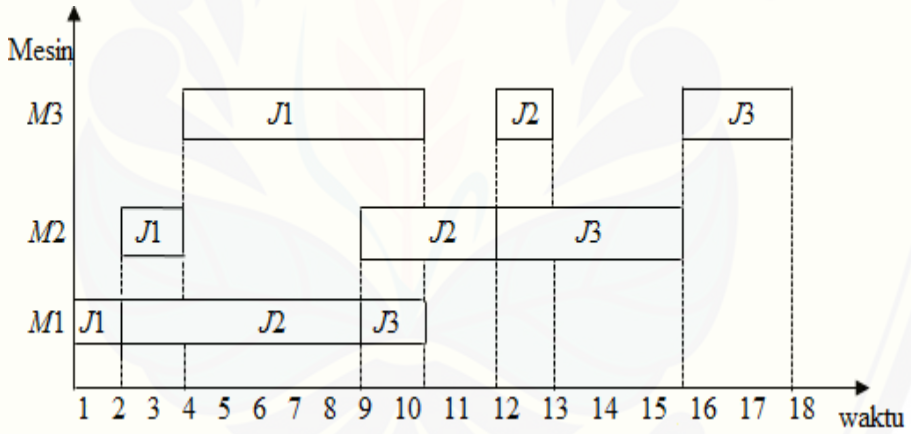
b. *General Flowshop* memiliki perbedaan dengan *pure flowshop,* jika *pure flowshop* mengerjakan *job* yang akan datang dikerjakan di semua mesin tetapi pada *general flowshop* beberapa *job* yang datang tidak harus dikerjakan disemua mesin.



Gambar 2.2 Pola aliran *General Flowshop*

2.3 Diagram *Gantt*

Diagram *Gantt* merupakan diagram perencanaan yang digunakan untuk penjadwalan sumber daya dan alokasi waktu (Heizer dan Render, 2006). Diagram *Gantt* adalah contoh teknik non matematis yang banyak digunakan dan sangat popular dikalangan para *manager* karena sederhana dan mudah dibaca. Diagram *Gantt* dikembangkan oleh Henry Laurence Gantt pada tahun 1910. Diagram *Gantt* merupakan salah satu alat yang berfungsi untuk merencakan penjadwalan dan memantau kegiatan pada suatu proyek. Diagram *Gantt* juga dapat dilihat urutan kegiatan ataupun tugas yang harus dilakukan berdasarkan prioritas waktu yang ditentukan. Diagram *Gantt* adalah grafik hubungan antara alokasi sumber daya dengan waktu. Pada sumbu vertikal digambarkan jenis sumber daya (mesin) yang digunakan dan sumbu horizontal digambarkan satuan waktu (Ginting, 2009). Diagram *gantt* dapat ditunjukan pada Gambar 2.3



Gambar 2.3 Diagram *Gantt*

Keuntungan menggunakan diagram *Gantt* sebagai berikut (Ginting, 2009):

a. Penggunaan diagram *gantt* memungkinkan evaluasi lebih awal mengenai penggunaan sumber daya seperti yang telah direncanakan.
b. Pekerjaan mudah diperiksa pada setiap waktu karena sudah tergambar jelas
c. Semua pekerjaan diperhatikan secara grafis dalam suatu diagram yang mudah dipahami.

2.4 Algoritma *Iterated Greedy*

Algoritma *Iterated Greedy* merupakan algoritma metaheuristik yang telah banyak diterapkan untuk permasalahan penjadwalan *job flowshop*. Prosedur kunci algoritma *Iterated Greedy* adalah destruktif dan konstruktif, dimana algoritma *Iterated Greedy* dimulai dari tahap destruktif dan dilanjutkan tahap konstruktif. Dalam tahap destruktif, dilakukan dengan menghilangkan beberapa komponen solusi secara acak dari solusi sebelumnya. Sedangkan pada tahap konstruktif, dilakukan dengan merekontruksi ulang solusi dengan mekanisme *greedy heuristic*. Setelah kandidat solusi telah selesai, maka akan ditentukan kriteria apakah solusi yang ditemukan dapat mengganti solusi sebelumnya. Iterasi *Iterated Greedy* akan berhenti sampai menemukan kriteria solusi yang ditentukan. Untuk menerapkan algoritma *Iterated Greedy* akan digunakan konstruktif *greedy heuristic* yang dikenal dengan metode NEH (Ribas *et al.*, 2011).

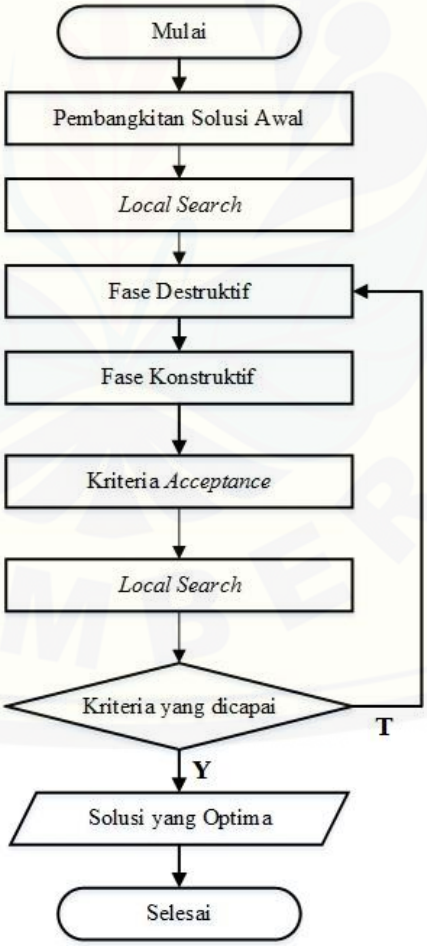
Langkah-langkah algoritma NEH adalah sebagai berikut (Ginting, 2009):

a. Menghitung waktu proses tiap *job*.
b. Mengurutkan *job* menurut jumlah waktu prosesnya dari yang terbesar hingga yang terkecil (*descending*).
c. Mengambil *job* yang menempati posisi ke-1 dan ke-2 pada daftar hasil pengurutan *job*.
d. Membuat 2 alternatif calon urutan parsial baru dan hitung *makespan* parsial dari calon urutan parsial baru.
e. Memilih calon urutan parsial baru yang memiliki *makespan* parsial yang terkecil.
f. Menambahkan satu *job* dari daftar pengurutan *job* yang menempati posisi ke-3.
g. Membuat 3 calon urutan parsial baru dengan memasukkan *job* yang diambil ke dalam setiap *slot* urutan parsial sebelumnya dan hitung *makespan* parsial dari calon urutan parsial baru.
h. Memilih calon urutan parsial baru yang memiliki *makespan* parsial terkecil.
i. Mengulangi langkah f-h untuk *job* dan posisi berikutnya hingga semua pekerjaan dalam daftar urutan *job* terjadwalkan.

Langkah-langkah algoritma *Greedy* sebagai berikut (Ruiz dan Stutzle, 2007):

a. Bangkitkan solusi awal (x_0).
b. Terapkan *local search* terhadap x_0, simpan sebagai x.
c. Ulangi langkah berikut sampai kriteria pemberhentian dicapai.
    1) Lakukan destruksi terhadap x, simpan sebagai x_p.
    2) Lakukan kontruksi terhadap x_p, simpan sebagai x_c.
    3) Terapkan *local search* terhadap x_c, simpan sebagai x^*.
    4) Lakukan *acceptance* pada solusi x, x_c, x^*, solusi terbaik disimpan sebagai x.

Pada umumnya prosedur dari algoritma *Iterated Greedy* akan ditunjukan pada Gambar 2.4 melalui berikut ini



Gambar 2.4 *Flowchart Iterated Greedy*

2.5 Algoritma *Social Cognitive Optimization*

Social Cognitive Optimization (SCO) merupakan algoritma optimasi metaheuristik berbasis populasi yang didasarkan pada teori *social cognitive*. Teori *social cognitive* mendefinisikan tingkah laku manusia yang sedang belajar dengan cara mengamati. Kunci pokok dari algoritma SCO adalah proses pembelajaran individu dari sekumpulan agen dengan memori dan pengetahuaanya dalam *social library* (Xie *et al*., 2002).

Teori *social cognitive* menunjukkan bahwa sebuah pikiran adalah kekuatan aktif yang membangun realitas seseorang, mengkodekan informasi secara selektif, melakukan perilaku atas dasar harapan, dan memberlakukan susunan pada tindakan sendiri. Melalui asas sebab-akibat, perilaku individu dibentuk oleh interaksi lingkungan dan kognisi seseorang, yang berubah dari waktu ke waktu sebagai fungsi kedewasaan dan pengalaman dengan pembelajaran melalui pengamatan model simbolik.

Simbol berfungsi sebagai mekanisme untuk berpikir meskipun perilaku melalui proses kognitif yang paling pengaruh. Melalui pembentukan simbol, seperti gambar atau bahasa, manusia dapat memberikan arti dan pendekatan pada pengalamannya. Kemampuan simbolik memungkinkan manusia untuk menyimpan pengetahuan dalam memorinya untuk memandu perilaku masa depan meskipun tidak ada model hidup yang tersedia. Hal ini menunjukkan bahwa manusia mampu untuk memodelkan perilaku yang diamati, yang berbeda dari *Swarm Intelligence*.

Simbol menyediakan mekanisme yang memungkinkan untuk memecahkan masalah kognitif dan terlibat dalam aksi *foresighted*, yang artinya seseorang dapat memikirkan akibat dari suatu perbuatan tanpa benar-benar melakukannya. Penelitian menunjukkan bahwa banyak pemikiran manusia berbasis linguistik, dan terdapat korelasi antara perkembangan kognitif dan akuisisi pengetahuan (Bandura, 1986).

Misalkan $f(x)$ adalah masalah optimasi global, x adalah sebuah kondisi ruang masalah S. Dalam SCO, setiap kondisi disebut titik pengetahuan, dan fungsi f adalah fungsi kebaikan. Di SCO, terdapat populasi agen kognitif (N_c) yang memecahkan masalah secara bersamaan, dengan *social sharing library*. Setiap agen

memegang memori pribadi yang berisi satu titik pengetahuan, dan *social sharing library* berisi himpunan titik pengetahuan (N_L). Algoritma berjalan dalam siklus pembelajaran yang berulang atau iteratife (T), yang mana perilaku sistem dalam siklus t hanya tergantung pada status sistem dalam siklus $t-1$. Aliran proses SCO dalam berikut:

a. Inisialisasi

Bangkitkan secara random titik pengetahuan $\mathbf{x}_i$ ($\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{id}\}$) dari setiap agen i dalam *social sharing library* X. Kemudian bentuk solusi diskrit $\mathbf{y}$

b. Siklus Pembelajaran

Pada setiap siklus t ($t = 1, \dots, T$), lakukan proses berikut:

1) Hitung Kualitas (*Fitness*)

Kualitas F_i setiap agen dihitung berdasarkan fungsi tujuan $f(\mathbf{y})$ dari masalah.

$$F_i = \begin{cases} f(\mathbf{y}), & \text{untuk kasus maksimasi} \\ \frac{1}{f(\mathbf{y})}, & \text{untuk kasus minimasi} \end{cases} \qquad (2.1)$$

2) Pembelajaran Observasi

Pembelajaran observasi dilakukan dengan mencari titik pengetahuan terbaik sebagai $\mathbf{x}_{Base}$ dan pilih titik pengetahuan lain secara random sebagai $\mathbf{x}_{Ref}$. Kemudian bandingkan kedua titik pengetahuan. Jika $\mathbf{x}_{Base}$ lebih baik dari $\mathbf{x}_{Ref}$, maka bentuk titik pengetahuan baru untuk menggantikan $\mathbf{x}_{Ref}$ menggunakan persamaan berikut:

$$\mathbf{x}' = \mathbf{x}_{Ref} + \alpha * Rand * (\mathbf{x}_{Base} - \mathbf{x}_{Ref}) \qquad (2.2)$$

dimana α dalah parameter ($\alpha > 0$) dan $Rand$ adalah vektor ($\{r_1, r_2, \dots, r_d\}$) dengan bilangan random (0,1).

3) Perbaikan *Library*

Perbaikan *library* dilakukan dengan mengganti titik pengetahuan terburuk dengan titik pengetahuan baru secara random. Titik pengetahuan terburuk dicari dengan,

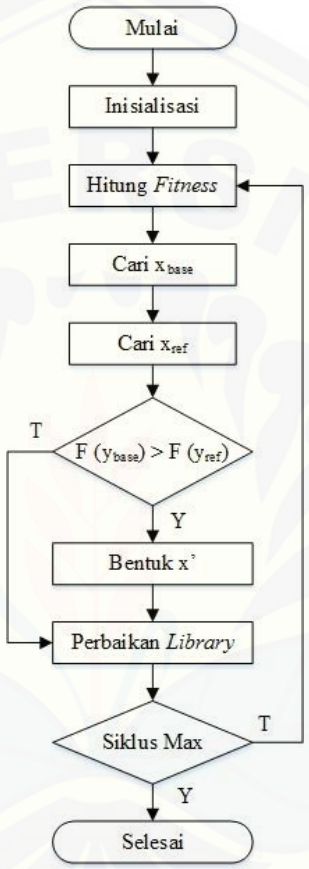
$$\mathbf{W} = \{i | \mathbf{F}_i < (\max(\mathbf{F}) - \min(\mathbf{F})) \cdot w + \min(\mathbf{F})\} \qquad (2.3)$$

dimana w adalah parameter *worst* (0,1).

c. Pemberhentian

Jika siklus t telah mencapai T, maka iterasi siklus dihentikan dan titik pengetahuan terbaik sebagai solusi (Xie *et al*., 2002).

Pada umumnya prosedur dari algoritma *Social Cognitive Optimization* adalah sebagai berikut ini:



Gambar 2.5 *Flowchart Social Cognitive Optimization*

2.6 Kriteria Kekonvergenan

Menurut Sivanandam dan Deepa (2008) pemberhentian atau kriteria kekonvergenan suatu algoritma dapat dituliskan sebagai berikut:

a. Batas Iterasi

Algoritma berhenti ketika batas iterasi sudah tercapai.

b. Nilai *fitness*/fungsi objektif tidak terjadi perubahan

Proses algoritma akan berhenti jika tidak ada perubahan pada nilai *fitness* atau fungsi nilai fungsi objektif terbaiknya.

BAB 3. METODE PENELITIAN

3.1 Data Penelitan

Data yang digunakan adalah data sekunder dari skripsi Laila (2014) dan data simulasi. Data pada skripsi Laila adalah data produksi sepatu "UD. Duta Kulit" menggunakan data tersebut terdiri dari 8 pekerjaan (*job*) *dan* 8 mesin (*machine*). Sedangkan data simulasi merupakan data yang diambil dari *website* http://mistic.heig-vd.ch/taillard/.../ordonnancement.html. Data simulasi ini dibuat oleh Taillard (1993). Data – data tersebut dapat dilihat pada Lampiran.

3.2 Langkah-langkah Penelitian

Langkah-langkah yang dilakukan pada penelitian ini, secara sistematis diuraikan sebagai berikut:

a. Studi Literatur

   Penelitiaan ini dimulai dengan mengumpulkan literatur dari berbagai sumber seperti jurnal, buku dan skripsi yang berkaitan dengan permasalahan penjadwalan *flowshop* dan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* yang dibahas. Studi literatur ini merupakan langkah awal untuk menemukan sumber-sumber yang dibutuhkan untuk penelitian ini.

b. Pengambilan Data

   Pada langkah ini yaitu melakukan pengambilan data sekunder dari skripsi Laila (2014) dan data simulasi merupakan data yang diambil dari *website*.

c. Pengolahan Data

   Menjadwalkan produksi *flowshop* menggunakan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* pada data yang telah dikumpulkan pada langkah b.

d. Pembuatan Program

   Pembuatan program menggunakan *software* MATLAB. Pada langkah ini akan dituliskan *script* program berdasarkan langkah kerja dari kedua algoritma yang digunakan.

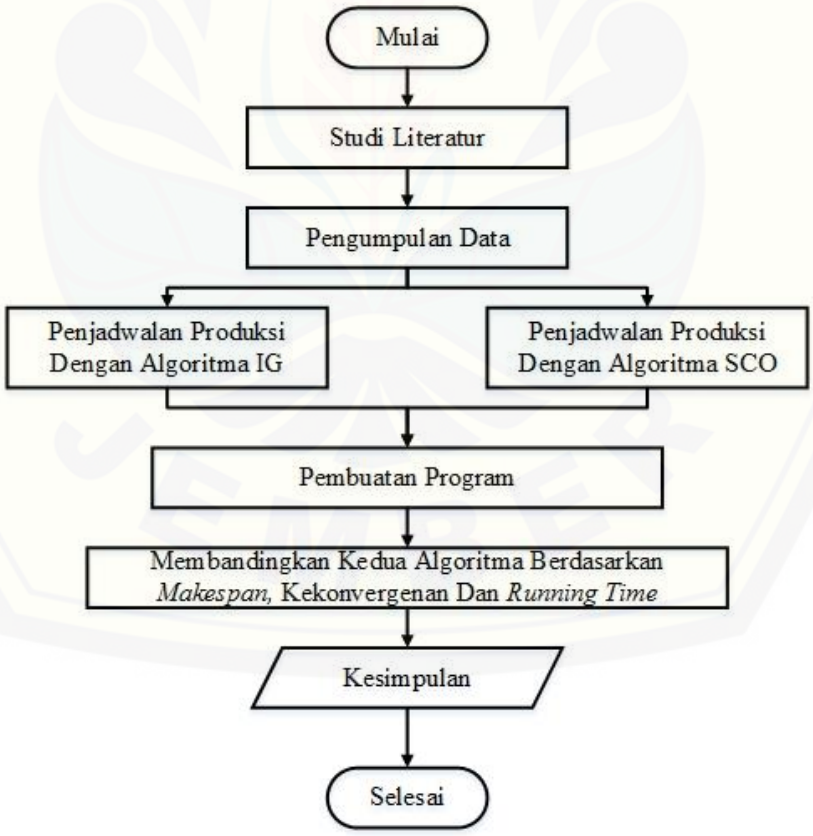
e. Perbandingan Algoritma

Membandingkan hasil dari kedua algoritma dengan menggunakan program yang telah dibuat berdasarkan nilai *makespan*, *running time* dan tingkat kecepatan kekonvergenan. Nilai *makespan* dan *running time* yang dipilih adalah nilai yang paling kecil. Sedangkan untuk tingkat kecepatan kekonvergenan algoritma dapat dilihat dengan melihat nilai *makespan* yang dihasilkan pada iterasi ke berapa data lebih cepat menunjukan konstan.

f. Kesimpulan

Pada langkah terakhir dilakukan penarikan kesimpulan dengan menjawab tujuan penelitian yang sebelumnya telah dibuat dan saran-saran untuk perbaiki bagi penelitian selanjutnya.

Keseluruhan langkah penelitian yang dilakukan secara sistematis dapat dilihat pada Gambar 3.1.



Gambar 3.1 Skema Penelitian

BAB 5. PENUTUP

5.1 Kesimpulan

Berdasarkan hasil percobaan yang dilakukan dengan menggunakan empat data berbeda dan masing-masing percobaan sebanyak sepuluh kali *running* untuk parameter yang sama yaitu $\alpha = 0{,}90$; $w = 0{,}37$ dan populasi= 20 pada algoritma *Social Cognitive Optimization* dapat diambil kesimpulan sebagai berikut:

a. Hasil paling optimal (*makespan* yang paling minimum) yang dihasilkan dari setiap data yang digunakan:
   1) Data 1 (8 *job*, 8 mesin) menghasilkan *makespan* = 1508 menit.
   2) Data 2 (20 *job*, 15 mesin) menghasilkan *makespan* = 1882 menit.
   3) Data 3 (30 *job*, 15 mesin) menghasilkan *makespan* = 2411 menit.
   4) Data 4 (50 *job*, 20 mesin) menghasilkan *makespan* = 4022 menit.

b. Ditinjau dari nilai *makespan*, algoritma *Iterated Greedy* menghasilkan nilai *makespan*-nya lebih optimal (minimum) daripada algoritma *Social Cognitive Optimization*. Hal ini dikarena algoritma *Iterated Greedy* memiliki langkah *Local Search* yang berfungsi untuk meningkatkan kualitas solusi yang dihasilkan, sedangkan pada algoritma *Social Cognitive Optimization* tidak memiliki langkah *Local Search*. Ditinjau dari tingkat kecepatan kekonvergenan, algoritma *Iterated Greedy* lebih cepat mencapai nilai kekonvergenan daripada algoritma *Social Cognitive Optimization*. Selain itu ditinjau dari *running time* algoritma *Iterated Greedy* menghasilkan waktu yang lebih singkat (cepat) daripada algoritma *Social Cognitive Optimization*.

5.2 Saran

Peneliti lebih lanjut dapat dilakukan untuk mengembangkan algoritma *Iterated Greedy* dan algoritma *Social Cognitive Optimization* dalam menyelesaikan masalah penjadwalan yaitu tentang minimasi *mean flowtime*, *mean tardiness* dan *weight tardiness*. Selain itu, masih terbuka untuk menerapkan algoritma *Iterated*

Greedy dan algoritma *Social Cognitive Optimization* pada masalah penjadwalan dengan multiobyektif.

DAFTAR PUSTAKA

Baker, K. 1974. *Introducing to sequencing and scheduling.* New York: John Wiley and Sons. Inc.

Bandura, A. 1986. *Social Foundations of Thought and Action: A Social Cognitive Theory*. New Jersey: Prentice-Hall.

Fang, H.L. 1994. *Genetic Algorithm in Timetabling and Scheduling.* Departement of Artifical Intelligence University of Edinburgh.

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

Heizer, J. dan Render, B. 2006. *Operations Management.* 7th Edition. Book 2. New Jersey: Pearson Education Inc. Prentice Hall.

Laila, I. 2014. Penerapan Algoritma *Simulated Annealing* dan Algoritma Tabu Search pada Produksi Sepatu dengan Penjadwalan *Flowshop*. Tidak dipublikasikan. Skripsi. Jember. FMIPA Universitas Jember.

Ribas, I., Companys, R. dan Tort-Martorell, X. 2011. An Iterated Greedy Algorithm for the Flowshop Scheduling Problem with Blocking. *Omega*, **39**(3): 293-301.

Ruiz, R., dan Stutzle, T. 2007. A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem. *European Journal of Operational Research*, **177**: 2033-2049.

Sivanandam, S. N. dan Deepa, S. N. 2008. *Introduction to Genetic Algorithm*. India: PSG College of Technology Coimbatore.

Xie, X. F., Zhang, W. J. dan Yang, Z. L. 2002. Social Cognitive Optimization for Nonlinear Programming Problems. *Proceedings of the First International Conference on Machine Learning and Cybernetics*, 779-783.

LAMPIRAN

Lampiran A. Data Penelitian

A.1 Data Produksi Sepatu UD. Duta Kulit

Job	Mesin							
	M1	M2	M3	M4	M5	M6	M7	M8
J1	72	36	48	100	48	120	24	54
J2	108	72	120	106	48	120	36	54
J3	84	48	72	102	48	120	30	54
J4	120	60	168	156	66	120	50	53
J5	138	74	180	172	66	126	48	54
J6	74	42	78	124	42	120	24	54
J7	84	44	96	150	48	120	42	54
J8	90	54	140	116	54	120	30	54

A.2 Data Simulasi 20 *Job* 15 Mesin

Job	MESIN														
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
J1	25	75	75	76	38	62	38	59	14	13	46	31	57	92	3
J2	67	5	11	11	40	34	77	42	35	96	22	55	21	29	16
J3	22	98	8	35	59	31	13	46	52	22	18	19	64	29	70
J4	99	42	2	35	11	92	88	97	21	56	17	43	27	19	23
J5	50	5	59	71	47	39	82	35	12	2	39	42	52	65	35
J6	48	57	5	2	60	64	86	3	51	26	34	39	45	63	54
J7	40	43	50	71	46	99	67	34	6	95	67	54	29	30	60
J8	59	3	85	6	46	49	5	82	18	71	48	79	62	65	76
J9	65	55	81	15	32	52	97	69	82	89	69	87	22	71	63
J10	70	74	52	94	14	81	24	14	32	39	67	59	18	77	50
J11	18	6	96	53	35	99	39	18	14	90	64	81	89	48	80
J12	44	75	12	13	74	59	71	75	30	93	26	30	84	91	93
J13	39	56	13	29	55	69	26	7	55	48	22	46	50	96	17
J14	57	14	8	13	95	53	78	24	92	90	68	87	43	75	94
J15	93	92	18	28	27	40	56	83	51	15	97	48	53	78	39
J16	47	34	42	28	11	11	30	14	10	4	20	92	19	59	28
J17	69	82	64	40	27	82	27	43	56	17	18	20	98	43	68
J18	84	26	87	61	95	23	88	89	49	84	12	51	3	44	20
J19	43	54	18	72	70	28	20	22	59	36	85	13	73	29	45
J20	7	97	4	22	74	45	62	95	66	14	40	23	79	34	8

A.3 Data Simulasi 30 *Job* 15 Mesin

Job	MESIN														
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
J1	99	43	6	99	23	98	84	24	30	53	34	95	50	48	38
J2	19	24	65	16	94	9	60	32	59	85	9	36	22	25	5
J3	54	62	93	78	59	71	49	88	40	13	17	88	47	30	56
J4	60	16	79	84	84	42	59	14	74	60	98	17	42	31	19
J5	49	52	46	50	1	14	2	56	64	51	75	28	9	37	6
J6	59	65	85	40	23	39	99	46	17	94	6	67	69	86	8
J7	10	7	22	36	31	75	57	49	44	21	77	70	64	46	69
J8	53	74	93	26	54	89	82	66	37	63	71	17	58	4	46
J9	76	72	42	17	27	56	78	5	72	19	90	46	43	56	17
J10	18	79	93	71	48	23	20	90	94	87	6	36	84	25	83
J11	52	61	45	60	15	74	49	26	94	54	1	58	56	54	72
J12	63	73	82	84	15	54	52	52	36	21	45	41	21	97	50
J13	90	90	77	33	31	26	14	75	92	70	55	56	39	49	23
J14	87	45	58	34	29	83	24	48	97	89	84	82	53	99	10
J15	35	32	30	93	58	28	88	16	98	4	82	98	26	29	77
J16	18	92	62	59	3	94	34	56	24	18	66	53	30	41	10
J17	2	26	17	18	60	39	23	95	81	56	34	8	47	72	56
J18	6	79	65	58	94	45	80	3	29	80	27	60	94	14	76
J19	31	79	87	79	57	48	33	42	93	86	54	32	8	16	63
J20	96	1	75	42	45	51	10	58	71	92	23	18	63	27	63
J21	84	82	16	61	43	75	28	15	19	93	22	1	62	9	5
J22	46	29	50	12	72	18	79	73	23	1	58	1	95	25	71

Job	MESIN														
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
J23	10	39	49	56	71	40	90	28	89	42	9	92	52	6	20
J24	70	63	68	97	86	81	38	7	53	48	43	59	88	29	87
J25	81	97	65	60	15	29	9	80	78	85	95	85	91	28	92
J26	39	6	59	34	34	32	12	7	35	4	53	69	89	3	40
J27	98	85	51	9	24	7	59	98	50	98	64	31	31	29	1
J28	59	68	3	8	2	9	69	14	72	84	69	54	45	59	7
J29	92	21	53	64	59	79	52	14	61	86	82	98	83	24	87
J30	51	70	94	80	35	56	8	94	11	3	60	73	26	21	45

A.4 Data Simulasi 50 *Job* 20 Mesin

Job	MESIN																			
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20
J1	48	40	54	71	52	70	41	76	52	24	5	43	68	10	49	9	81	30	93	17
J2	85	18	54	42	41	71	68	82	54	49	21	1	58	1	69	58	40	59	66	29
J3	33	34	77	42	95	2	71	73	19	25	45	88	19	40	42	17	81	72	70	67
J4	51	41	74	97	26	4	25	12	17	76	6	79	49	39	1	27	44	75	1	18
J5	22	99	7	7	72	24	19	81	23	72	50	95	31	67	67	22	12	28	68	88
J6	52	51	44	38	64	11	62	20	54	15	83	79	55	48	38	37	42	81	89	60
J7	82	43	57	1	89	11	41	50	68	2	4	65	20	56	46	36	33	56	13	50
J8	45	11	63	59	69	39	44	61	67	72	74	59	16	26	90	66	56	47	95	39
J9	92	2	88	90	45	88	90	94	34	1	81	64	70	55	7	33	21	35	62	61
J10	89	21	61	18	77	20	42	59	79	12	56	14	21	43	89	31	71	92	47	71

Job	MESIN																			
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20
J11	61	84	3	73	35	36	79	88	54	96	22	70	10	4	76	40	85	84	93	65
J12	68	72	74	97	63	33	96	4	63	31	1	98	39	65	72	20	7	63	33	26
J13	41	65	34	71	19	49	87	61	79	61	29	22	74	68	60	23	82	33	94	42
J14	17	40	40	28	6	62	83	95	44	91	79	39	68	79	1	20	96	62	62	70
J15	39	89	37	7	84	60	61	73	3	75	3	75	3	48	74	67	39	32	69	25
J16	9	83	30	3	31	93	86	49	34	91	56	80	33	37	35	63	72	46	22	73
J17	21	46	33	54	22	64	20	76	77	97	28	54	81	95	81	72	80	75	18	81
J18	52	30	38	70	22	15	66	26	55	34	13	65	87	38	85	89	77	22	67	44
J19	63	95	18	94	73	51	35	57	38	65	69	60	90	68	32	40	11	75	97	51
J20	68	37	39	13	76	77	6	6	53	41	72	71	46	24	46	50	12	39	92	54
J21	93	95	8	27	53	75	3	42	5	24	73	88	57	20	99	39	74	75	44	24
J22	83	14	66	96	11	36	20	5	72	38	79	10	27	27	90	8	83	10	61	69
J23	22	56	54	50	51	9	15	36	20	79	51	84	40	59	48	27	65	44	40	83
J24	5	75	43	17	10	92	22	36	7	71	77	70	10	24	78	77	56	42	16	48
J25	37	96	81	12	92	86	63	88	28	57	58	23	4	95	80	12	82	53	5	57
J26	58	59	65	78	68	50	38	97	72	94	59	42	5	19	27	54	69	2	56	51
J27	4	7	36	35	80	95	51	59	93	5	61	4	43	30	93	76	42	99	30	46
J28	88	75	81	40	61	94	78	24	19	44	96	23	90	94	80	97	24	44	64	52
J29	5	99	60	87	64	36	78	32	4	18	26	87	74	26	90	45	35	54	27	23
J30	93	95	11	14	99	86	41	26	50	74	21	6	67	87	46	84	11	89	89	66
J31	50	71	71	5	60	29	17	29	98	61	87	58	6	60	84	92	23	25	23	57
J32	75	60	77	48	87	52	98	8	55	97	55	68	59	90	50	98	57	43	72	35
J33	46	22	11	49	34	30	79	72	77	47	55	63	58	89	71	94	95	13	97	46

Job	MESIN																			
	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	M16	M17	M18	M19	M20
J34	25	98	71	68	8	72	57	39	83	17	90	31	81	6	97	98	82	82	52	82
J35	42	77	71	19	80	31	66	90	18	15	76	58	92	34	66	8	65	67	84	42
J36	41	42	69	81	95	16	45	52	48	35	72	80	81	4	3	4	96	53	14	80
J37	6	6	12	86	26	52	70	93	81	31	89	99	99	71	74	7	43	86	1	93
J38	44	54	36	40	68	49	45	58	44	65	72	65	53	48	90	98	60	7	27	48
J39	9	16	56	27	50	57	55	87	44	47	29	82	80	43	75	10	70	38	28	2
J40	29	91	85	51	86	34	73	12	14	51	1	38	74	92	60	43	36	23	82	30
J41	21	97	4	85	21	55	34	62	78	11	34	17	3	43	38	44	45	17	3	83
J42	29	6	45	15	60	29	97	91	13	8	50	46	72	86	7	30	28	13	27	42
J43	38	10	93	6	72	38	73	88	44	66	79	47	61	6	64	18	2	6	91	37
J44	21	20	51	96	51	42	52	37	85	18	44	60	68	3	6	20	81	96	30	9
J45	16	54	53	57	46	84	1	76	26	7	69	88	29	73	32	51	4	74	75	75
J46	27	54	90	25	97	68	14	54	29	14	8	1	60	13	16	41	81	35	18	79
J47	56	7	31	55	85	35	82	63	35	54	52	77	82	94	81	25	24	56	23	79
J48	33	50	22	70	59	51	80	84	47	88	27	18	34	47	4	41	56	42	26	66
J49	31	83	9	34	62	83	61	41	58	96	87	18	56	2	95	21	51	13	31	96
J50	62	95	8	3	27	19	36	97	87	62	86	21	37	11	11	67	84	34	48	97

Lampiran B

B.1 *Script* Proses

```
%% reset
set(handles.listbox1,'string','');
set(handles.listbox2,'string','');
cla(handles.axes1,'reset');
axes(handles.axes1);
set(handles.axes1,'xlim',[0 1000],'xtick',[0 200 400 600 800
1000],...
    'ylim',[0 500],'ytick',[0 100 200 300 400 500]);
grid on;
xlabel('Iterasi');ylabel('Makespan');
pause(0.000001);
data=get(handles.uitable1,'UserData');
imax=str2num(get(handles.vimax,'string'));
popX=str2num(get(handles.vx,'string'));
alpha=str2num(get(handles.valpha,'string'));
T=str2num(get(handles.vimax,'string'));
w=str2num(get(handles.vw,'string'));
if ~isempty(data) && ~isempty(imax) && ~isempty(popX) &&
~isempty(alpha) && ...
        ~isempty(T) && ~isempty(w)
tic;
%% proses IG
[m n]=size(data);
x0=randperm(m);
x=localsearch(x0);
makespan1=hitungmakespan(x,data);
for i=1:imax
    %Destruction
    e=ceil(rand*m);
    ps=ceil(rand*(m-e+1));
    xs=x(ps:ps+e-1);
    %Construction
    xn=NEH(xs,data);
    xc=x;
    xc(ps:ps+e-1)=xn;
    makespan1a=hitungmakespan(xc,data);
    %Local Search
    xaksen=localsearch(xc);
    makespan1b=hitungmakespan(xaksen,data);
    if makespan1a<makespan1b
        if makespan1a<makespan1
            x=xc;
            makespan1=makespan1a;
        end
    else
        if makespan1b<makespan1
            x=xaksen;
            makespan1=makespan1b;
        end
    end
    konver1(i)=makespan1;
end
[makespan1,waktu]=hitungmakespan(x,data);
```

```
conver1=find(konver1==min(konver1));
tampil1='Urutan Job: ';
for i=1:m-1
    tampil1=[tampil1 num2str(x(i)) '-'];
end
tampil1=[tampil1 num2str(x(m))];
hasil1={'Hasil Algoritma Iterated Greedy:';'';...
    tampil1;'';['Makespan: ' num2str(makespan1)];'';...
    ['Algoritma konvergen pada iterasi ke-'
num2str(conver1(1))];'';...
    ['Waktu komputasi: ' num2str(toc)]};
set(handles.listbox1,'string',char(hasil1),'UserData',{konver1,x,w
aktu});
%% proses SCO
tic;
[m n]=size(data);
lb=-1;
ub=1;
%x=importdata('xawal.txt');
for i=1:popX
    X(i,:)=rand(1,m)*(ub-lb)+lb;
    Y(i,:)=diskrit(X(i,:));
    makespan2(i)=hitungmakespan(Y(i,:),data);
    Fitness(i)=1/makespan2(i);
end
for t=1:T
    for i=1:popX-1
        for j=i+1:popX
        if Fitness(i)>Fitness(j)
            X(j,:)=combinex(X(i,:),X(j,:),alpha);
            Y(j,:)=diskrit(X(j,:));
            makespan2(j)=hitungmakespan(Y(j,:),data);
            Fitness(j)=1/makespan2(j);
        end
        end
    end
    worst=find(Fitness<(max(Fitness)-
min(Fitness))*w+min(Fitness));
    if ~isempty(worst)
        for i=1:length(worst)
            X(worst(i),:)=X(worst(i),:)+alpha*(rand(1,m)*2-1);
            Y(worst(i),:)=diskrit(X(worst(i),:));

makespan2(worst(i))=hitungmakespan(Y(worst(i),:),data);
            Fitness(worst(i))=1/makespan2(worst(i));
        end
    else
        worst=1:ceil(popX*w);
        for i=1:ceil(popX*w)
            X(worst(i),:)=X(worst(i),:)+alpha*(rand(1,m)*2-1);
            Y(worst(i),:)=diskrit(X(worst(i),:));

makespan2(worst(i))=hitungmakespan(Y(worst(i),:),data);
            Fitness(worst(i))=1/makespan2(worst(i));
        end
    end
```

```
    konver2(t)=min(makespan2);
end
best=find(makespan2==min(makespan2));
solusi=Y(best(1),:);
[mspan,waktu]=hitungmakespan(solusi,data);
conver2=find(konver2==min(konver2));
tampil2='Urutan Job: ';
for i=1:m-1
    tampil2=[tampil2 num2str(solusi(i)) '-'];
end
tampil2=[tampil2 num2str(solusi(m))];
hasil2={'Hasil Algoritma Social Cogn. Optimization:';'';...
    tampil2;'';['Makespan: ' num2str(min(makespan2))];'';...
    ['Algoritma konvergen pada iterasi ke-'
num2str(conver2(1))];'';...
    ['Waktu komputasi: ' num2str(toc)]};
set(handles.listbox2,'string',char(hasil2),'UserData',{konver2,sol
usi,waktu});
    min1=find(konver1==min(konver1));
    min2=find(konver2==min(konver2));
    axes(handles.axes1);
    plot(konver1,'r','LineWidth',2);
    hold on;
    plot(konver2,'b','LineWidth',2);
    grid on;
    legend('IG','SCO');ymin=min([min(konver1) min(konver2)]);
    ymax=max([max(konver1) max(konver2)]);
    ylim([ymin-(ymax-ymin)/8 ymax+(ymax-ymin)/16]);
    xlabel('Iterasi'); ylabel('Makespan');

line(min1(1),min(konver1),'Marker','s','MarkerEdgeColor','k','Mark
erFaceColor','b','MarkerSize',8);

line(min2(1),min(konver2),'Marker','s','MarkerEdgeColor','k','Mark
erFaceColor','b','MarkerSize',8);
else
    errordlg('Data atau Parameter kosong');
end
```

B.2 *Script* Diagram *gantt*

```
function gantt(waktu,flow)
[m n]=size(waktu);
color=rand(m,3);
figure,
for i=2:n
    for j=1:m
        if j==1
            patch([waktu(j,i-1) waktu(j,i) waktu(j,i) waktu(j,i-
1)],[i-2 i-2 i-1 i-1],color(j,:));
        else
            patch([max(waktu(j,i-1),waktu(j-1,i)) waktu(j,i)
waktu(j,i) max(waktu(j,i-1),waktu(j-1,i))],[i-2 i-2 i-1 i-
1],color(j,:));
        end
        hold on;
```

```
    end
    text(-max(max(waktu))/8+1,i-1.5,['M' num2str(i-
1)],'FontSize',12);
end
axis([-max(max(waktu))/8 max(max(waktu))+max(max(waktu))/3 0 n-
0.5]);
set(gca,'YTick',[]);
for i=1:m
    ket{i}=['Job ' num2str(flow(i))];
end
legend(char(ket));
xlabel('Waktu'); ylabel('Mesin');
```