PERBANDINGAN ALGORITMA *ARTIFICIAL BEE COLONY* DAN ALGORITMA *DIFFERENTIAL EVOLUTION PLUS* UNTUK PENJADWALAN *FLOWSHOP*

SKRIPSI

Oleh

Nurhalimatus Sa'dia
NIM 111810101018

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

PERBANDINGAN ALGORITMA *ARTIFICIAL BEE COLONY* DAN ALGORITMA *DIFFERENTIAL EVOLUTION PLUS* UNTUK PENJADWALAN *FLOWSHOP*

SKRIPSI

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

Oleh

Nurhalimatus Sa'dia
NIM 111810101018

JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS JEMBER
2015

PERSEMBAHAN

Skripsi ini saya persembahkan untuk:

1. Ayahanda Saneri dan Ibunda Misri yang telah memberikan kasih sayang, doa dan bimbingan dalam perjalanan hidupku;
2. Adikku Moh. Arifin dan 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. Almamater Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember, SMAN 1 Kraksaan, SMP Negeri 1 Maron, SDN 1 Wonorejo, dan TK Dharma Wanita.

MOTTO

Sesungguhnya Allah tidak merubah keadaan suatu kaum
sehingga mereka merubah keadaan yang ada pada diri mereka sendiri.
(terjemahan Surat Ar-Rad Ayat 11)[*)]

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

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Nurhalimatus Sa'dia

NIM : 111810101018

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Perbandingan Algoritma *Artificial Bee Colony* dan Algoritma *Differential Evolution Plus* untuk Penjadwalan *Flowshop*" adalah benar-benar 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, 25 Juni 2015

Yang menyatakan,

Nurhalimatus Sa'dia

NIM 111810101018

SKRIPSI

PERBANDINGAN ALGORITMA *ARTIFICIAL BEE COLONY* DAN ALGORITMA *DIFFERENTIAL EVOLUTION PLUS* UNTUK PENJADWALAN *FLOWSHOP*

Oleh

Nurhalimatus Sa'dia
NIM 111810101018

Pembimbing

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

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

PENGESAHAN

Skripsi berjudul "Perbandingan Algoritma *Artificial Bee Colony* dan Algoritma *Differential Evolution Plus* untuk Penjadwalan *Flowshop*" telah diuji dan disahkan pada:

Hari, tanggal :

Tempat : Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember

Tim penguji:

Ketua,	Sekretaris,
Ahmad Kamsyakawuni, S.Si., M.Kom	Kusbudiono, S.Si., M.Si
NIP. 197211291998021001	NIP. 197704302005011001
Anggota I,	Anggota II,
M. Ziaul Arif, S.Si., M.Sc	Dian Anggraeni, S.Si., M.Si
NIP. 198501112008121002	NIP. 198202162006042002

Mengesahkan

Dekan,

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

NIP 196101081986021001

RINGKASAN

Perbandingan Algoritma *Artificial Bee Colony* dan Algoritma *Differential Evolution Plus* untuk Penjadwalan *Flowshop*; Nurhalimatus Sa'dia, 111810101018; 2015: 52 halaman; Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Penjadwalan merupakan suatu proses dalam perencanaan dan pengendalian produksi serta pengalokasian sumber daya pada suatu waktu tertentu dengan memperhatikan kapasitas sumber daya yang ada, sehingga diperlukan adanya pengaturan sumber daya yang ada secara efisien. Adapun tujuan dari penjadwalan produksi yaitu meminimasi *makespan* (total waktu yang dibutuhkan untuk menyelesaikan seluruh pekerjaan). Penjadwalan *flowshop* merupakan salah satu jenis penjadwalan dimana setiap *job* akan diproses mengalir pada satu arah yaitu melalui urutan mesin yang sama.

Data yang digunakan dalam penelitian ini adalah data pada skripsi Nurjanah (2015), yaitu data pembuatan kerupuk UD. Samjaya. Dalam proses pembuatan kerupuk terdapat 10 pekerjaan yang diproses pada 9 mesin dengan setiap pekerjaan diproses tepat satu kali pada setiap mesin. Setiap pembuatan jenis kerupuk yang diproduksi melewati proses dengan urutan yang sama. Industri ini belum memiliki jadwal yang tetap dalam proses produksinya. Penumpukan pekerjaan untuk diproses dan sistem penjadwalan produksi yang kurang tepat dapat menyebabkan keterlambatan dalam menyelesaikan produksi, sehingga terjadi keterlambatan dalam pengiriman barang.

Pada penelitian ini menggunakan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* yang akan diterapkan pada penjadwalan

pembuatan kerupuk UD. Samjaya. Kedua algoritma akan dibandingkan untuk menyelesaikan permasalahan penjadwalan produksi *flowshop* dengan tujuan mencari solusi terbaik berdasarkan nilai *makespan* minimum dan tingkat kecepatan kekonvergenan.

Penelitian ini dilakukan melalui beberapa langkah, yaitu pengambilan data kemudian melakukan penjadwalan menggunakan kedua algoritma. Langkah selanjutnya adalah menentukan tingkat kecepatan kekonvergenan dari masing-masing algoritma. Kemudian membandingkan hasil dari kedua algoritma berdasarkan nilai *makespan* dan tingkat kecepatan kekonvergenan sehingga dapat diambil kesimpulan dari perbandingan tersebut.

Berdasarkan hasil penelitian yang dilakukan pada kedua algoritma dengan sembilan kali pengujian diperoleh nilai *makespan* yang sama yaitu 8.878 menit. Akan tetapi urutan jadwal yang dihasilkan pada setiap pengujian adalah berbeda. Hal ini menunjukkan bahwa algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* dapat dikatakan efektif untuk diterapkan pada penjadwalan produksi kerupuk. Apabila ditinjau dari tingkat konvergensinya, pada algoritma *Artificial Bee Colony* rata-rata konvergen pada iterasi ke-4 sedangkan pada algoritma *Differential Evolution Plus* rata-rata konvergen pada iterasi ke-1. Sehingga algoritma *Differential Evolution Plus* lebih cepat konvergen dibandingkan dengan algoritma *Artificial Bee Colony.*

PRAKATA

Puji syukur kehadirat Allah SWT. atas segala rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Perbandingan Algoritma *Artificial Bee Colony* dan Algoritma *Differential Evolution Plus* untuk Penjadwalan *Flowshop*". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata satu (S1) pada Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Jember.

Pada kesempatan ini penulis menyampaikan terima kasih kepada semua pihak yang telah membantu penyusunan skripsi ini diantaranya:

1. Prof. Drs. Kusno, DEA, Ph.D. selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam yang telah mengesahkan penulisan skripsi ini;
2. Ahmad Kamsyakawuni, S.Si., M.Kom. selaku Dosen Pembimbing Utama dan Kusbudiono, S.Si., M.Si. selaku Dosen Pembimbing Anggota yang telah memberikan bimbingan serta arahan selama penulisan skripsi ini sehingga dapat terselesaikan dengan baik;
3. M. Ziaul Arif, S.Si., M.Sc. dan Dian Anggraeni, S.Si., M.Si. selaku Dosen Penguji yang telah memberikan kritik dan saran sehingga skripsi ini menjadi lebih baik lagi;
4. seluruh dosen dan karyawan jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam;
5. Ayahanda Saneri dan Ibunda Misri yang selalu memberikan semangat serta doa demi terselesaikannya skripsi ini;

6. Adikku Moh. Arifin dan keluarga besar yang selalu memberikan dukungan dalam segala hal;
7. sahabatku Diana, Emil dan Fia yang selalu memberi motivasi dan menemani masa-masa penulisan skripsi ini;
8. keluarga besar REMPONGERS (Riprodh, Endod, Mprol, Piyo, Onyil, Nitong, Gephong, Encrong, Rimboy, Sokep) yang selalu memberi dukungan pada saat penyelesaian skripsi ini;
9. keluarga besar KRAMAT'11 yang selalu memberi dukungan tanpa henti;
10. rekan kerjaku maz Train dan mbk Arista yang telah membantu dan memberi dukungan dalam penyelesaian skripsi ini;
11. semua pihak yang tidak dapat saya sebutkan satu per satu.

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

Jember, Juni 2015 Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

Halaman

DAFTAR GAMBAR

Halaman

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan dunia usaha yang semakin pesat menimbulkan banyaknya jenis usaha yang bermunculan sehingga menyebabkan terjadinya persaingan yang kompetitif diantara perusahaan tersebut. Dengan adanya persaingan tersebut mendorong perusahaan untuk melakukan berbagai strategi dalam menjalankan usahanya. Langkah awal untuk menghadapi persaingan dan menjadi strategi perusahaan yaitu harus mampu memenuhi permintaan konsumen dengan tepat waktu. Oleh karena itu, setiap perusahaan perlu melakukan penjadwalan produksi yang baik.

Penjadwalan produksi suatu perusahaan merupakan hal terpenting demi kelangsungan dan kelancaran produksi suatu perusahaan dalam memenuhi permintaan konsumennya. Ketepatan waktu dalam proses produksi dilakukan dengan cara meminimalisasi total waktu yang dibutuhkan sehingga hasil produksinya selesai tepat waktu dan juga mengoptimalkan penggunaan mesin dalam menyelesaikan seluruh pekerjaan supaya dapat mengurangi waktu mesin yang menganggur. Dengan adanya penjadwalan yang optimum diharapkan dapat menyelesaikan pekerjaan dengan tepat waktu.

Salah satu permasalahan penjadwalan produksi yang sering terjadi adalah penjadwalan *flowshop*. Penjadwalan produksi *flowshop* merupakan suatu proses produksi dari masing-masing *job* yang mempunyai urutan proses produksi dengan melalui mesin yang sama. Tujuan utama dari penjadwalan yaitu untuk menentukan urutan jadwal pekerjaan yang mampu meminimalkan nilai *makespan*. Nilai *makespan* adalah total waktu yang dibutuhkan untuk menyelesaikan seluruh *job.*

Beberapa penelitian sebelumnya tentang penjadwalan *flowshop*, dilakukan oleh Sugioko (2013) yang menganalisa performa algoritma *Bee Colony* yang dimodifikasi dengan menggunakan operasi *swap* dan *tabu list* pada penjadwalan

flowshop. Pada kajian tersebut menunjukkan bahwa algoritma *Bee Colony* yang telah dimodifikasi atau algoritma *Artificial Bee Colony* menghasilkan nilai *makespan* yang lebih unggul daripada algoritma *Bee Colony*. Pada penelitian lain, dilakukan oleh Sari (2015) yang mengkaji tentang perbandingan algoritma *Simulated Annealing* dengan algoritma *Differential Evolution Plus* untuk penjadwalan *flowshop*. Pada kajian tersebut menyatakan bahwa algoritma *Differential Evolution Plus* memiliki tingkat kekonvergenan yang lebih baik dibandingkan dengan algoritma *Simulated Annealing*.

Data yang digunakan dalam penelitian ini adalah data pada skripsi Nurjanah (2015), yaitu data pembuatan kerupuk UD. Samjaya yang terletak di Jalan Galunggung, Desa Klatakan, Kecamatan Tanggul, Kabupaten Jember. Dalam proses pembuatan kerupuk terdapat 10 pekerjaan yang diproses pada 9 mesin dengan setiap pekerjaan diproses tepat satu kali pada setiap mesin. Setiap pembuatan jenis kerupuk yang diproduksi melewati proses dengan urutan yang sama. Industri ini belum memiliki jadwal yang tetap dalam proses produksinya. Penumpukan pekerjaan untuk diproses dan sistem penjadwalan produksi yang kurang tepat dapat menyebabkan keterlambatan dalam menyelesaikan produksi, sehingga terjadi keterlambatan dalam pengiriman barang.

Berdasarkan penjelasan diatas, penulis meneliti perbandingan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* dalam memperoleh urutan *job* yang diterapkan pada penjadwalan pembuatan kerupuk UD. Samjaya. Kemudian menentukan algoritma yang memiliki performa lebih baik, yang ditinjau dari aspek nilai *makespan* dan tingkat kecepatan kekonvergenan diantara kedua algoritma tersebut.

1.2 Rumusan Masalah

Permasalahan yang akan dibahas dalam skripsi ini adalah sebagai berikut:

a. Bagaimana penerapan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* untuk penjadwalan *flowshop*?

b. Bagaimana perbandingan kedua algoritma tersebut ditinjau berdasarkan aspek *makespan* dan tingkat kecepatan kekonvergenan?

1.3 Batasan Masalah

Batasan-batasan masalah yang terdapat dalam skripsi ini adalah sebagai berikut:

a. Bahan baku yang dibutuhkan selalu tersedia;
b. Setiap *job* memiliki *ready time* yang sama;
c. Mesin berjalan dengan normal, tidak ada gangguan;
d. Waktu pada saat membawa *job* ke mesin lainnya diabaikan.

1.4 Tujuan

Tujuan dari penulisan skripsi ini adalah sebagai berikut:

a. Menerapkan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* untuk penjadwalan produksi *flowshop*;
b. Mengetahui hasil dari perbandingan kedua algoritma tersebut yang ditinjau dari aspek *makespan* dan tingkat kecepatan kekonvergenan.

1.5 Manfaat

Manfaat yang dapat diperoleh dari penulisan skripsi ini adalah memberikan informasi kepada pembaca tentang algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* serta perbedaan kedua algoritma tersebut dalam menyelesaikan penjadwalan *flowshop*.

BAB 2. TINJAUAN PUSTAKA

2.1 Penjadwalan

Penjadwalan menurut Conway adalah proses pengurutan pembuatan produk secara menyeluruh pada beberapa mesin. Penjadwalan juga didefinisikan oleh Vollman sebagai rencana pengaturan urutan kerja serta pengalokasian sumber, baik waktu maupun fasilitas untuk setiap operasi yang harus diselesaikan. Sedangkan menurut Baker suatu penjadwalan didefinisikan sebagai proses pengalokasian sumber daya untuk memilih sekumpulan tugas dalam jangka waktu tertentu. Keputusan yang dibuat dalam sebuah penjadwalan meliputi pengurutan pekerjaan (*sequencing*), waktu mulai dan selesai pekerjaan (*timing*), dan urutan operasi untuk suatu pekerjaan (*routing*) (Ginting, 2009).

Menurut Baker (1974) tujuan dari penjadwalan yaitu sebagai berikut:

a. Mengurangi persediaan barang setengah jadi atau mengurangi sejumlah pekerjaan yang menunggu dalam antrian mesin yang masih mengerjakan tugas yang lain;
b. Meningkatkan produktivitas mesin dengan mengurangi waktu tunggunya;
c. Mengurangi keterlambatan pada pekerjaan yang mempunyai batas waktu penyelesaian sehingga akan meminimasi biaya kelambatan.
d. Membantu pengambilan keputusan mengenai waktu dimana suatu produk harus selesai diproses atau masih diproduksi.

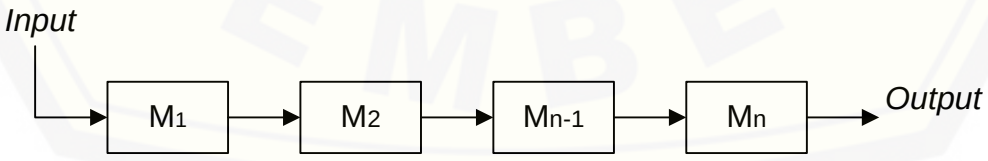
Secara garis besar, penjadwalan dibedakan menjadi penjadwalan *flowshop* dan penjadwalan *jobshop*. Permasalahan yang membedakan antara penjadwalan *flowshop* dan penjadwalan *jobshop* yaitu berdasarkan pola aliran kerja yang tidak memiliki tahapan-tahapan proses yang sama. Untuk melakukan suatu penjadwalan diperlukan data yang diantaranya adalah jenis dan banyaknya *job* yang akan diproses, urutan ketergantungan antar proses produksinya, waktu proses untuk masing-masing operasi,

serta banyaknya mesin yang dibutuhkan pada setiap operasi. Setelah semua data terkumpulkan maka proses penjadwalan dapat dilakukan.

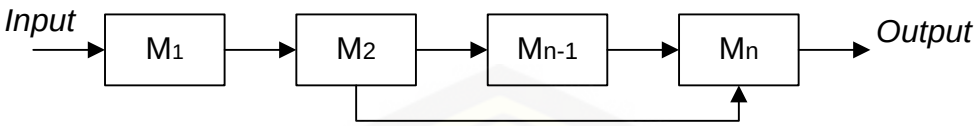
2.2 Penjadwalan *Flowshop*

Penjadwalan *flowshop* adalah proses penentuan urutan pekerjaan yang memiliki lintasan 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 di mesin akhir dan jumlah tahapan proses umumnya sama dengan jumlah jenis mesin yang digunakan. Pada umumnya sistem produksi yang bersifat *flowshop*, terdiri dari beberapa mesin (m) dan memiliki 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. Masing-masing *job* memiliki waktu proses pada masing-masing mesin. Tujuan penjadwalan pada umumnya adalah menemukan suatu urutan *job* yang bertujuan untuk meminimumkan *makespan* (Sari, 2015).

Setiap *job* dapat diperlakukan seolah-olah *job* tersebut memiliki *m* operasi yang tetap. Aliran pekerjaan *flowshop* terbagi menjadi 2 yaitu *pure flowshop* dan *general flowshop*. Pada aliran pekerjaan *pure flowshop*, setiap *job* memiliki satu operasi pada setiap mesin yang dapat dilihat seperti pada Gambar 2.1. Sedangkan pada *general flowshop* suatu pekerjaan dimungkinkan terdiri kurang dari *m* operasi dengan operasi pada mesin-mesin yang tidak berdekatan (bersebelahan) dan operasi terakhir tidak selalu dimulai pada mesin 1 dan diakhiri pada mesin *m* yang dapat dilihat pada Gambar 2.2.



Gambar 2.1 Aliran *pure flowshop*



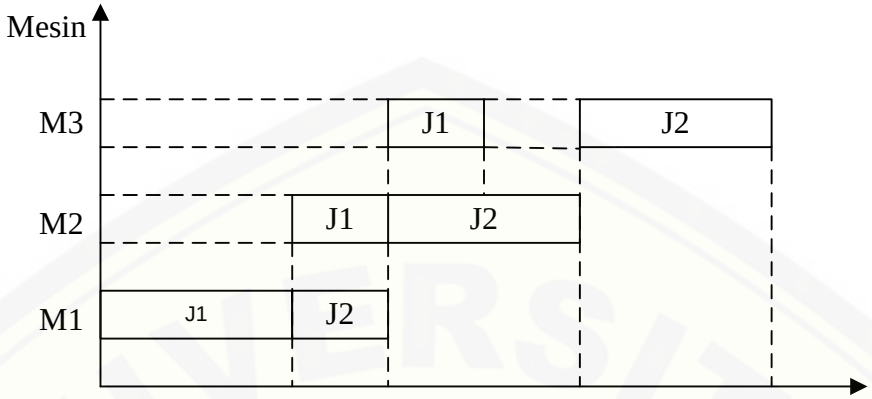
Gambar 2.2 Aliran *general flowshop*

Karakteristik dasar penjadwalan *flowshop* yaitu (Kuncoro, 2013):

a. Terdapat *n job* yang tersedia dan siap diproses pada waktu $t = 0$.
b. Waktu *set up independent* terhadap urutan pengerjaan.
c. Terdapat *m* mesin berbeda yang tersedia secara *continue*.
d. Operasi-operasi individual tidak dapat dipecah-pecah.

2.3 Diagram *Gantt*

Masalah penjadwalan sebenarnya masalah murni pengalokasian dan dengan bantuan model matematis akan dapat ditentukan solusi optimal. Model-model penjadwalan akan memberikan rumusan masalah yang sistematik berikut dengan solusi yang diharapkan. Sebagai alat bantu yang digunakan dalam menyelesaikan masalah penjadwalan dikenal satu model yang sederhana dan umum digunakan secara luas yakni diagram *Gantt* (*gantt chart*). *Gantt chart* pertama kali diperkenalkan oleh Henry Laurence Gantt pada tahun 1916. *Gantt chart* merupakan grafik hubungan antara alokasi sumber daya dengan waktu. Pada sumbu vertikal digambarkan jenis sumber daya yang digunakan dan sumbu horizontal digambarkan satuan waktu. *Gantt chart* dapat ditunjukkan pada Gambar 2.3.



Gambar 2.3 Diagram *gantt*

Keuntungan menggunakan diagram *gantt* adalah (Ginting, 2009):

a. Dalam situasi keterbatasan sumber, penggunaan diagram *gantt* memungkinkan evaluasi lebih awal mengenai penggunaan sumber daya seperti yang telah direncanakan;
b. Kemajuan pekerjaan mudah diperiksa pada setiap waktu karena sudah tergambar dengan jelas;
c. Semua pekerjaan diperlihatkan secara grafis dalam suatu diagram yang mudah dipahami.

2.4 Algoritma

Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis. Algoritma ditulis dengan notasi khusus yang mudah dimengerti dan notasi dapat diterjemahkan menjadi suatu bahasa pemrograman. Algoritma akan memerlukan masukan (*input*) tertentu untuk memulainya dan akan menghasilkan keluaran (*output*) tertentu pada akhirnya. Hal-hal yang perlu diperhatikan dalam algoritma adalah mencari langkah-langkah yang paling sesuai untuk penyelesaian suatu masalah, karena setiap algoritma memiliki karakteristik tertentu yang memiliki kelebihan dan kekurangan (Nugraha, 2012). Suatu algoritma

harus menghasilkan *output* yang efektif dalam waktu yang relatif singkat dan penggunaan memori yang relatif sedikit dengan langkah yang berhingga dan prosesnya berakhir baik dalam keadaan diperoleh suatu solusi ataupun tidak sesuai dengan harapan.

2.5 Algoritma *Artificial Bee Colony*

Algoritma *Artificial Bee Colony* pertama kali dikenalkan oleh Karaboga pada tahun 2005, yaitu berdasarkan perilaku kawanan lebah dalam mencari makanan serta cara lebah berkomunikasi untuk memberikan informasi kepada lebah lainnya tentang letak sumber makanan. Ketika beberapa ekor lebah menemukan makanan maka mereka akan mengundang lebah lainnya melalui tarian. Informasi mengenai sumber makanan terdiri dari tiga hal yakni arah, jarak dari sarang, dan kualitas (jumlah nektar). Semakin bagus kualitas sumber makanan maka semakin lama durasi tarian yang dilakukan sehingga semakin banyak lebah yang mengikuti (Suyanto, 2010).

Dalam koloni lebah, terdapat tiga tipe lebah yaitu lebah pekerja (*Employed Bee)* bertugas untuk mencari sumber makanan (*food source*) dan menginformasikan tentang letak sumber makanan kepada lebah penjaga. Lebah penjaga *(Onlooker Bee)* bertugas menyimpan makanan dan menjaga sarang saat lebah pekerja mencari sumber makanan, serta bertugas untuk menentukan jalur untuk mendapatkan sumber makanan sesuai dengan informasi yang disarankan oleh lebah pekerja. Sedangkan lebah pengintai *(Scout Bee)* bertugas untuk mengikuti jalur yang disarankan oleh lebah penjaga untuk menemukan sumber makanan (Chong *et al*, 2006).

Performa algoritma *Artificial Bee Colony* dipengaruhi oleh penentuan parameter yang akan digunakan yaitu jumlah solusi dan jumlah iterasi. Karena kedua parameter tersebut memberikan efek yang signifikan terhadap performa algoritma *Artificial Bee Colony*. Semakin besar jumlah solusi dan jumlah iterasinya maka dapat menghasilkan kinerja yang lebih baik karena memiliki ruang pencarian yang semakin besar. Sehingga menghasilkan *makespan* yang selalu optimal dan cepat konvergen (Sugioko, 2013).

Langkah-langkah pada algoritma *Artificial Bee Colony* adalah sebagai berikut:

a. Inisialisasi Solusi Awal

Penentuan parameter dilakukan terlebih dahulu sebelum memulai suatu generasi (iterasi). Parameter-parameter tersebut yaitu ukuran jumlah populasi lebah, jumlah lebah pengintai, dan panjang *list* solusi yang akan digunakan, serta kriteria berhenti yaitu jumlah iterasi yang dipakai. Inisialisasi solusi awal menggunakan solusi yang diperoleh secara acak dengan persamaan (2.1).

$$x_{ij} = x_{j\,min} + rand(0,1).\left(x_{j\,max} - x_{j\,min}\right) \quad (2.1)$$

dengan,

x_{ij} = inisialisasi kemungkinan solusi ke-*i* dengan parameter ke-*j*

$x_{j\,min}$ = nilai kemungkinan solusi terkecil berdasarkan parameter *j*

$x_{j\,max}$ = nilai kemungkinan solusi terbesar berdasarkan parameter *j*

$rand(0,1)$ = nilai *random* antara 0 sampai 1

i = 1, 2, …, *SN* dengan *SN* adalah jumlah kemungkinan solusi (sumber makanan)

j = 1, 2, …, *D* dengan *D* adalah jumlah parameter yang digunakan.

b. Menentukan Solusi Alternatif

Tahap penentuan solusi alternatif sering dikenal dengan istilah tahap lebah pekerja (*Employed Bee Phase*). Pada tahap ini dilakukan berdasarkan hasil dari inisialisasi solusi awal yang akan dijadikan acuan sejumlah *n* lebah untuk melakukan pencarian sumber nektar, sehingga didapatkan sejumlah *n* solusi alternatif dengan menggunakan persamaan (2.2).

$$v_{ij} = x_{ij} + \emptyset_{ij}\left(x_{kj} - x_{ij}\right) \quad (2.2)$$

dengan,

v_{ij} = nilai perluasan kemungkinan solusi ke-*i* dengan parameter *j*

x_{ij} = nilai kemungkinan solusi ke-*i* dengan parameter *j*

x_{kj} = nilai kemungkinan solusi ke-*k* dengan parameter *j*

k = 1, 2, …, *SN*

$\emptyset_{ij}$ = bilangan *random* antara -1 sampai 1

Setelah setiap kemungkinan solusi diperluas, akan diaplikasikan *greedy selection* antara nilai kemungkinan solusi x_{ij} dengan nilai hasil perluasan yaitu v_{ij}. Jika nilai v_{ij} lebih kecil dari nilai x_{ij} maka nilai v_{ij} tersebut akan dianggap sama dengan nilai x_{ij} dan nilai percobaan tetap bernilai 0. Jika tidak, nilai x_{ij} yang disimpan dan nilai percobaan ke-*i* ditambah dengan 1.

c. Evaluasi Populasi Awal

Setelah setiap kemungkinan solusi diperluas dan dibandingkan dengan nilai awal inisialisasi, tahap selanjutnya adalah menghitung kualitas dari setiap kemungkinan solusi menggunakan fungsi *fitness* sebagai berikut:

$$fitness\ (x_i) = \begin{cases} \dfrac{1}{(1+f(x_i))}, f(x_i) \geq 0 \\ 1 + |f(x_i)|, f(x_i) < 0 \end{cases} \quad (2.3)$$

d. Evaluasi Populasi Alternatif

Evaluasi populasi alternatif atau yang sering disebut dengan istilah tahap lebah penjaga (*Onlooker Bee Phase*) merupakan suatu tahap untuk menghitung nilai *probability* pada setiap kemungkinan solusi dengan menggunakan persamaan (2.4).

$$p_i = \frac{fitness_i}{\sum_{i=1}^{SN} fitness_i} \quad (2.4)$$

dengan,

p_i = nilai *probability*

$fitness_i$ = nilai *fitnesss* solusi ke-*i*

$\sum_{i=1}^{SN} fitness_i$ = jumlah nilai *fitnesss* ke-*i* sampai SN

e. Evaluasi Populasi Akhir

Tahapan evaluasi populasi akhir yang sering disebut dengan istilah pengintaian (*Scout Bee Phase*) merupakan tahapan evaluasi solusi secara keseluruhan. Pada tahap ini, akan diaplikasikan sebuah metode *rooellete-whele* yaitu memilih bilangan real secara *random* antara [0, 1] untuk setiap kemungkinan solusi. Jika

nilai p_i lebih besar dari bilangan *random* yang ditentukan, maka akan memperluas kembali kemungkinan solusi yang terpilih tersebut sesuai lebah pekerja sebelumnya.

f. Kriteria Berhenti

Kriteria pemberhentian yang digunakan untuk kasus ini adalah jumlah iterasi yang telah ditentukan pada awal perhitungan. Jika jumlah iterasi belum terpenuhi maka kembali pada langkah b.

2.6 Algoritma *Differential Evolution Plus*

Algoritma *Differential Evolution* (DE) pertamakali diperkenalkan oleh Storn dan Price pada tahun 1995 yang merupakan salah satu metode metaheuristik (Santosa dan Willy, 2011). Secara umum, proses pencarian solusi pada algoritma *Differential Evolution* yaitu melalui tahapan-tahapan berupa inisialisasi populasi, mutasi, *crossover*, dan seleksi. Namun pada penelitian ini menggunakan algoritma *Differential Evolution Plus* yang merupakan pengembangan dari algoritma DE murni dengan cara melakukan tiga modifikasi. Pertama, memodifikasi pengendalian nilai parameter *F* dan *Cr* (*adaptive parameters*) dengan cara menghitung parameter *F* dan *Cr* pada setiap generasi menggunakan formula tertentu sehingga nilai parameter *F* dan *Cr* pada setiap generasi berubah-ubah. Kedua, menambahkan prosedur *local search* pada DE untuk meningkatkan kualitas solusi yang dihasilkan. Ketiga, *crossover* pada mulanya merupakan langkah untuk menyilangkan populasi target dengan populasi mutan, dimodifikasi menjadi langkah untuk memilih populasi target yang akan dimutasi sehingga dilakukan sebelum proses mutasi. Langkah ini betujuan untuk mengurangi waktu komputasi (Wiratno *et al*., 2012).

Performa algoritma *Differential Evolution Plus* dipengaruhi oleh penentuan parameter yang akan digunakan yaitu jumlah populasi dan jumlah iterasi. Semakin besar jumlah populasi dan iterasinya maka dapat menghasilkan kinerja yang lebih

baik karena memiliki ruang pencarian yang semakin besar. Sehingga menghasilkan *makespan* yang selalu optimal dan cepat konvergen (Wiratno *et al*., 2012).

Langkah-langkah algoritma *Differential Evolution Plus* adalah sebagai berikut:

a. Inisialisasi populasi

Tahap inisialisasi populasi pada algoritma DE *plus* memiliki proses yang sama dengan algoritma DE murni. Sebelum melakukan inisialisasi terhadap titik populasi maka perlu dilakukan penentuan batas atas (*ub*) dan batas bawah (*lb*). Untuk pembangkitan nilai awal generasi $g = 0$, variabel ke-j dan vektor ke-i bisa diwakili dengan persamaan (2.5).

$$x_{j,i,0} = lb_j + rand_j(0,1)\left(ub_j - lb_j\right) \tag{2.5}$$

dengan,

x = individu target

j = variabel job j

i = populasi ke-i

lb = batas bawah

ub = batas atas

Bilangan *random* dibangkitkan dengan fungsi *rand*(), dimana bilangan yang dihasilkan terletak antara (0,1). Solusi awal diperoleh dengan cara mengurutkan populasi menggunakan prosedur SPV (*Smallest Position Value*). SPV merupakan prosedur pengurutan nilai x tiap populasi dari nilai terkecil ke nilai terbesar.

b. Crossover

Langkah *crossover* pada algoritma DE *plus*, dilakukan sebelum mutasi untuk memilih populasi yang akan dimutasi. Konsep mutasi yaitu bila bilangan radom lebih kecil atau sama dengan *Cr* maka populasi tersebut akan dimutasi, bila bilangan random lebih besar dari *Cr* maka populasi itu tidak dimutasi (Wiratno *et al*., 2012). Probabilitas *crossover*, $Cr \in (0,1)$ adalah nilai yang didefinisikan

untuk mengendalikan fraksi nilai parameter yang disalin dari mutan. Probabilitas *crossover* untuk tiap generasi akan ditentukan dengan persamaan berikut:

$$Cr = Cr_{min} + G\frac{Cr_{max} - Cr_{min}}{MAXGEN} \quad (2.6)$$

dengan,

Cr = probabilitas *crossover*

Cr_{min} = nilai terkecil dari probabilitas *crossover*

Cr_{max} = nilai terbesar dari probabilitas *crossover*

G = iterasi pada saat waktu *trunning time*

$MAXGEN$ = jumlah maksimum iterasi

Tujuan dari penentuan nilai *Cr* adalah untuk meningkatkan keragaman vektor yang akan mengalami *crossover* dan menghindar dari *local optimal*.

c. Mutasi

Mutasi pada algoritma DE murni dan algoritma DE *plus* dilakukan dengan menggunakan formula yang sama. Algoritma DE *plus* melakukan mutasi dan kombinasi terhadap populasi yang telah dipilih pada langkah *crossover*. Mutasi dilakukan dengan cara menambahkan perbedaan dua vektor terhadap vektor ketiga secara acak. Formulasinya dapat dilihat pada persamaan (2.7).

$$v_{i,g} = x_{r0,g} + F\left(x_{r1,g} - x_{r2,g}\right) \quad (2.7)$$

dengan,

F = parameter mutasi

$x_{r0,g}, x_{r1,g}, x_{r2,g}$ = vektor acak

Faktor skala $F \in (0, 1)$ adalah bilangan *real* positif yang mengendalikan tingkat pertumbuhan populasi. Dalam langkah ini nilai parameter *F* tiap generasi akan berubah-ubah dengan menghitung nilai parameter pada tiap generasi dengan formula pada persamaan (2.8).

$$F = \begin{cases} \max\left(F_{min}, 1 - \left|\frac{f_{max}}{f_{min}}\right|\right) \text{ jika } \left|\frac{f_{max}}{f_{min}}\right| < 1 \\ \max\left(F_{min}, 1 - \left|\frac{f_{min}}{f_{max}}\right|\right) \text{ jika yang lain} \end{cases} \quad (2.8)$$

dengan,

f_{min} = nilai minimum fungsi objektif

f_{max} = nilai maksimum fungsi objektif

F_{min} = input parameter yang memastikan $f \in [F_{min}, 1]$

d. Seleksi

Tahap seleksi pada algoritma DE murni dan algoritma DE *plus* akan diperoleh solusi terbaik. Seleksi tersebut dapat ditunjukkan pada persamaan (2.9).

$$x_{i,g+1} = \begin{cases} v_{i,g}, & \text{jika } f(v_{i,g}) \leq f(x_{i,g}) \\ x_{i,g}, & \text{jika sebaliknya} \end{cases} \quad (2.9)$$

e. Local Search

Hasil dari seleksi akan dikenai prosedur *insert-based local search* yang cenderung mengarahkan pencarian ke daerah solusi yang menjanjikan dalam waktu relatif singkat. Prosedur dari *insert-based local search* adalah sebagai berikut:

1) Ubah nilai individual $x_i(t)$ menjadi permutasi π_{i_0} ;
2) Pilih secara acak *u* dan *v*, dimana $u \neq v$; $\pi_i = insert(\pi_{i\_0}, u, v)$;
3) $Set\ loop = 1$;

   Pilih secara acak *u* dan *v*, dimana $u \neq v$;

   $\pi_{i\_1} = insert(\pi_i, u, v)$;

   Jika $f(\pi_{i\_1}) < f(\pi_i)$, maka $\pi_i = \pi_{i\_1}$;

   $loop^{++}$;

   $while\ loop < (n \times (n-1))$.
4) Jika $f(\pi_i) < f(\pi_{i\_0})$, maka $\pi_{i\_0} = \pi_i$.

   dengan,

   $\pi_{i\_0}, \pi_i, \pi_{i\_1}$ = permutasi *job*

u = *job* yang dipilih secara acak

v = posisi

f. Mengecek iterasi maksimal

Penentuan iterasi maksimal digunakan untuk memutuskan apakah iterasi selesai atau tidak. Jika iterasi selesai, maka didapatkan solusi optimal. Namun jika tidak kembali pada langkah b.

2.7 Kriteria Kekonvergenan

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

a. Iterasi maksimum

Suatu algoritma akan berhenti ketika mencapai iterasi maksimum yang telah ditentukan.

b. Batas waktu maksimum

Proses algoritma akan berhenti ketika batas waktu maksimum telah terlampaui. Jika iterasi maksimum sudah tercapai sebelum batas waktunya terpenuhi maka prosesnya akan berhenti.

c. Nilai fungsi objektif tidak mengalami perubahan

Proses algoritma akan berhenti jika tidak ada perubahan pada nilai fungsi objektif yang optimal.

BAB 3. METODE PENELITIAN

3.1 Data Penelitian

Data yang akan digunakan dalam penelitian ini adalah data pada skripsi Nurjanah (2015), yaitu data pembuatan kerupuk UD. Samjaya yang terletak di Jalan Galunggung, Desa Klatakan, Kecamatan Tanggul, Kabupaten Jember. Produk yang dihasilkan oleh UD. Samjaya terdapat sepuluh jenis kerupuk. Pembuatannya dilakukan dengan sembilan proses (mesin), yaitu persiapan bahan, pengadonan, pelembutan, pencetakan, pengukusan, pendinginan, pemotongan (pembentukan), penjemuran dan pengemasan. Data yang diambil adalah data yang berupa waktu proses pembuatan kerupuk pada setiap mesin dalam satuan waktu yaitu menit. Data tersebut disajikan dalam Tabel 3.1.

Tabel 3.1 Data waktu proses pembuatan kerupuk (menit)

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9
J_1	15	160	15	30	30	1440	150	180	30
J_2	15	160	15	30	30	1440	180	360	30
J_3	15	160	15	30	30	360	85	180	30
J_4	15	60	30	10	3	360	10	360	30
J_5	15	60	30	10	3	300	30	180	30
J_6	15	60	30	10	3	300	10	360	30
J_7	15	60	30	10	3	720	30	180	30
J_8	15	45	15	35	30	720	72	360	10
J_9	15	45	15	35	30	1440	168	180	15
J_{10}	15	75	36	25	17	1440	45	360	25

J_1 = Kerupuk impala panjang
J_2 = Kerupuk impala pendek
J_3 = Kerupuk rajang
J_4 = Kerupuk mawar 2 udang besar
J_5 = Kerupuk mawar 2 udang kecil
J_6 = Kerupuk mawar SI besar
J_7 = Kerupuk mawar SI kecil
J_8 = Kerupuk kasandra besar
J_9 = Kerupuk kasandra kecil
J_{10} = Kerupuk obar-abir

M_1 = Persiapan bahan
M_2 = Pengadonan
M_3 = Mesin mulen (pelembutan)
M_4 = Pencetakan
M_5 = Pengukusan
M_6 = Pendinginan
M_7 = Pemotongan (pembentukan)
M_8 = Penjemuran
M_9 = Pengemasan

3.2 Langkah-langkah Penelitian

Langkah-langkah yang akan dilakukan dalam menyelesaikan permasalahan penjadwalan *flowshop* adalah sebagai berikut:

a. Studi literatur yang dilakukan pada penelitian ini adalah mempelajari mengenai penjadwalan, penjadwalan *flowshop*, *gantt chart*, algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus*;
b. Pengambilan data dilakukan dengan mengambil data yang ada pada skripsi Nurjanah (2015);
c. Menerapkan algoritma *Artificial Bee Colony* untuk penyelesaian masalah penjadwalan *flowshop*;
    1) Penentuan nilai parameter yang akan digunakan yaitu jumlah populasi lebah (iterasi), jumlah lebah pengintai (solusi), dan panjang *list* solusi. Akan tetapi, parameter yang akan diujikan adalah jumlah solusi dan iterasi yaitu dengan melakukan sembilan percobaan.
    2) Inisialisasi solusi ini terlebih dahulu melakukan pembangkitan solusi secara *random*, kemudian dilakukan *swap operator* yaitu dengan menukar 2 *job* pada setiap solusi dan menghitung nilai *makespan* dari masing-masing hasil *swap*

operator sehingga didapatkan nilai $x_{j\,max}$ dan $x_{j\,min}$. Setelah itu, melakukan inisialisasi solusi awal dengan menggunakan persamaan (2.1).

3) Menentukan solusi alternatif berdasarkan hasil dari inisialisasi solusi awal dengan menggunakan persamaan (2.2). Setelah setiap kemungkinan solusi diperluas, akan diaplikasikan *greedy selection* antara nilai kemungkinan solusi x_{ij} dengan nilai hasil perluasan yaitu v_{ij}. Jika nilai v_{ij} lebih kecil dari nilai x_{ij} maka nilai v_{ij} tersebut akan dianggap sama dengan nilai x_{ij} dan nilai percobaan tetap bernilai 0. Jika tidak, nilai x_{ij} yang disimpan dan nilai percobaan ke-*i* ditambah dengan 1.
4) Evaluasi populasi awal dengan menghitung nilai *fitness* pada setiap kemungkinan solusi dengan menggunakan persamaan (2.3).
5) Evaluasi populasi alternatif dengan menghitung nilai *probability* pada setiap kemungkinan solusi dengan menggunakan persamaan (2.4).
6) Evaluasi populasi akhir untuk mencari nilai terbaik dari semua kemungkinan solusi. Pada tahap ini, akan diaplikasikan sebuah metode *rooellete-whele* yaitu memilih bilangan real secara *random* antara [0, 1] untuk setiap kemungkinan solusi. Jika nilai p_i lebih besar dari bilangan *random* yang ditentukan, maka akan memperluas kembali kemungkinan solusi yang terpilih menggunakan persamaan (2.2). Setelah kemungkinan solusi terpilih diperluas, akan diaplikasikan *greedy selection* antara nilai kemungkinan solusi x_{ij} dengan nilai baru hasil perluasan yaitu v_{ij}. Kemudian diperoleh nilai percobaan yang maksimum, maka nilai kemungkinan solusi dengan nilai percobaan maksimum tersebut yang akan disimpan sebagai salah satu pilihan solusi terbaik dalam *list* solusi. Setelah itu proses kembali ke langkah 3 sampai kriteria pemberhentian yaitu sampai jumlah iterasi terpenuhi.

d. Menerapkan algoritma *Differential Evolution Plus* untuk penyelesaian masalah penjadwalan *flowshop*;

1) Penentuan nilai parameter yang akan digunakan yaitu populasi yang dibangkitkan, batas atas (*Ub*), batas bawah (Lb), nilai terkecil probabilitas *crossover* (Cr_{min}), nilai terbesar probabilitas *crossover* (Cr_{max}), nilai minimum parameter mutasi (F_{min}), dan jumlah iterasi maksimum (*MAXGEN*). Akan tetapi, parameter yang akan diujikan adalah jumlah populasi dan iterasi yaitu dengan melakukan sembilan percobaan.

2) Inisialisasi ini menghitung nilai individu pada tiap populasi (x_i) yang dibangkitkan menggunakan persamaan (2.5). Untuk mendapatkan solusi awal, maka nilai individu pada tiap populasi diurutkan menggunakan prosedur SPV. Selanjutnya menghitung nilai fungsi objektif (*makespan*) pada tiap populasi dan dievaluasi untuk menentukan f_{min} dan f_{max} yang merupakan nilai minimum dan maksimum fungsi objektif.

3) C*rossover* digunakan untuk menentukan populasi yang akan dimutasi atau tidak. Sebelumnya telah ditentukan nilai parameter Cr_{min}, Cr_{max}, dan *MAXGEN*. Kemudian dihitung nilai *crossover* (*Cr*) dengan persamaan (2.6). Setelah itu dibangkitkan bilangan *random* pada tiap populasi, apabila bilangan *random* $\leq Cr$ maka individu pada populasi ke – i akan dimutasi, apabila bilangan *random* $> Cr$ maka individu pada populasi ke–*i* tidak dimutasi.

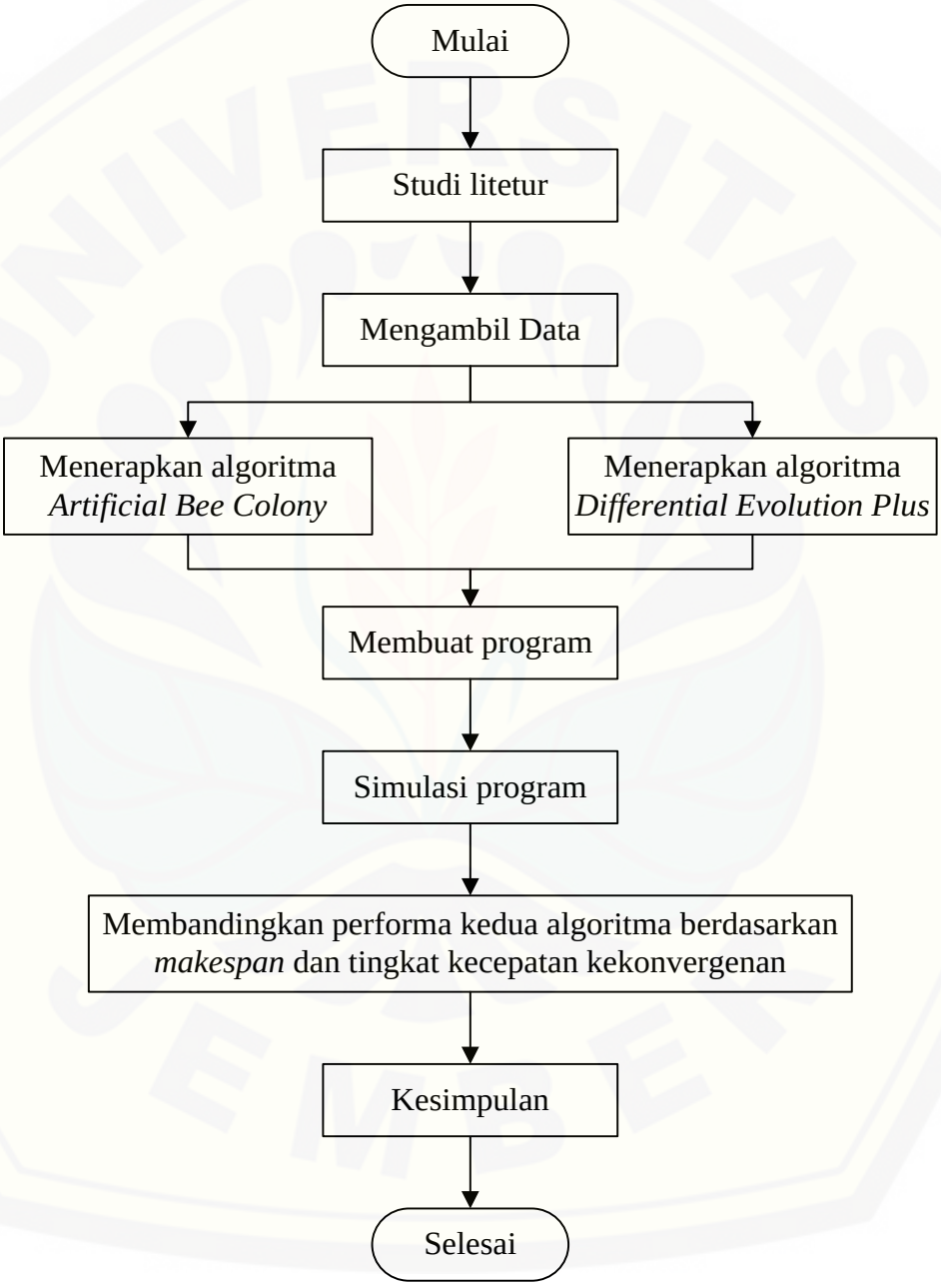
4) Mutasi dilakukan setelah menghitung nilai parameter mutasi (*F*) dengan menggunakan persamaan (2.8). Kemudian dilakukan mutasi dengan persamaan (2.7) dimana $x_{r0,g}, x_{r1,g}, x_{r2,g}$ pada persamaan tersebut merupakan nilai vektor yang dipilih secara acak dari nilai x_i pada tahap inisialisasi populasi sehingga diperoleh hasil mutasi pada populasi ke-i (v_i). Cek hasil mutasi apakah diluar batas atas dan batas bawah. Jika hasil mutasi $>$ batas atas maka akan diganti dengan 1, jika hasil mutasi $<$ batas bawah maka akan diganti dengan 0, sedangkan jika hasil mutasi bernilai antara $0-1$ maka

nilainya tetap. Kemudian hasilnya dlakukan prosedur SPV (*Smallest Position Value*) dan buat pula urutan permutasi *job*.

5) Seleksi dilakukan dengan membandingkan nilai *makespan* hasil mutasi $f(v_i)$ dengan nilai *makespan* awal $f(x_i)$ sehingga diperoleh nilai *makespan* yang minimum untuk tiap populasi ke-i $(f(x_i))$. Jika $f(v_i) < f(x_i)$ maka $x_i = v_i$, jika tidak maka $x_i = x_i$.
6) *Local search* ini dilakukan menggunakan prosedur *insert-based local search* yaitu prosedur dengan menyisipkan *job* ke-*u* pada posisi *v* dimana $u \neq v$. Pada tahap seleksi, telah diperoleh urutan permutasi *job* beserta nilai objektif, kemudian hasil seleksi tersebut disebut $f(\pi_{i\_0})$ yang merupakan nilai *makespan* permutasi *job* untuk populasi ke–*i* tahap *local search* ke–0. Pilih secara acak *job* ke-*u* dan posisi *v* dimana $u \neq v$, selanjutnya sisipkan *job u* ke posisi *v* pada setiap populasi serta hitung nilai *makespan* yang disebut dengan $f(\pi_i)$ yaitu nilai *makespan* permutasi *job* untuk populasi ke-*i*. Setelah itu dilakukan *looping* dengan penyisipan *job u* ke posisi *v* dalam $f(\pi_i)$ yaitu menghasilkan permutasi *job* $f(\pi_{i\_1})$. Bandingkan $f(\pi_i)$ dan $f(\pi_{i\_1})$, urutan *job* dengan nilai *makespan* lebih baik akan terpilih. Langkah tersebut dilakukan hingga kriteria pemberhentian *looping* yaitu $< n(n-1)$. Setelah *looping* dilakukan hingga kriteria pemberhentian, maka diperoleh urutan *job* beserta nilai *makespan* yang disebut $f(\pi_i)$ dari hasil *looping*. Bandingkan $f(\pi_i)$ dari hasil *looping* dengan $f(\pi_{i\_0})$ pada tahap seleksi sehingga akan diperoleh urutan *job*, nilai *makespan* dan nilai x_i.
7) Untuk iterasi selanjutnya, kembali pada proses langkah 2 hingga iterasi maksimal terpenuhi dan akan diperoleh urutan *job* yang optimal dengan nilai *makespan* minimum.

e. Langkah penelitian selanjutnya adalah pembutan program dari kedua algoritma dengan menggunakan *software Matlab*;

f. Membandingkan performa dari kedua algoritma menggunakan program yang telah dibuat berdasarkan nilai *makespan* dan tingkat kecepatan kekonvergenan;
g. Membuat kesimpulan berdasarkan hasil pada langkah f yang telah dilakukan.



Gambar 3.1 Skema langkah-langkah penelitian

BAB 4. HASIL DAN PEMBAHASAN

Pada bab ini akan dibahas mengenai perbandingan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* pada penjadwalan *flowshop* yang diterapkan pada produksi kerupuk UD. Samjaya. Perbandingan kedua algoritma tersebut dilakukan perhitungan dengan menggunakan bantuan *software* MATLAB.

4.1 Penyelesaian Penjadwalan *Flowshop* Secara Manual

Untuk mengetahui langkah kerja algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* dalam menyelesaikan masalah penjadwalan *flowshop*, maka dilakukan perhitungan secara manual dengan menggunakan data kecil. Data tersebut berupa data waktu proses produksi empat *job* pada tiga mesin yang dapat dilihat pada Tabel 4.1. Penggunaan data kecil ini bertujuan untuk memudahkan dalam memahami langkah kerja kedua algoritma tersebut.

Tabel 4.1 Waktu proses produksi (menit)

	M_1	M_2	M_3
J_1	67	125	12
J_2	73	118	19
J_3	93	115	11
J_4	91	100	18

4.1.1 Penyelesaian Manual dengan Algoritma *Artificial Bee Colony*

Berikut ini penyelesaian penjadwalan *flowshop* secara manual dalam satu iterasi menggunakan algoritma *Artificial Bee Colony*. Namun sebelumnya harus menentukan nilai parameter awalnya terlebih dahulu yaitu jumlah lebah pengintai

(solusi) = 3, panjang *list* solusi = 3 dan jumlah populasi lebah (iterasi) = 100. Langkah penyelesaiannya adalah sebagai berikut:

a. Inisialisasi Solusi Awal

Pada tahap ini dilakukan pembangkitan solusi secara *random*. Kemudian dihitung nilai *makespan* pada solusi yang dibangkitkan. Misalkan solusi yang dibangkitkan secara *random* yaitu $J_1 - J_2 - J_3 - J_4$. Maka solusi tersebut memiliki nilai *makespan* seperti ditunjukkan pada Tabel 4.2.

Tabel 4.2 Perhitungan *makespan* solusi awal

	M_1		M_2		M_3	
	S	E	S	E	S	E
J_1	0	67	67	192	192	204
J_2	67	140	192	310	310	329
J_3	140	233	310	425	425	436
J_4	233	324	425	525	525	543

S = *Start* atau mulai

E = *End* atau berhenti

Berdasarkan Tabel 4.2 tersebut, dapat diketahui bahwa nilai *makespan* dari solusi awal adalah 543 menit. Kemudian pada solusi awal tersebut dilakukan metode *Swap Operator (SO),* yaitu dengan cara melakukan pertukaran antara 2 *job* yang dilakukan secara *random*. Metode *Swap Operator (SO)* dilakukan sebanyak jumlah solusi yang akan digunakan yaitu sebanyak 3 kali. Hasil yang didapat dari pertukaran *job* menggunakan metode *Swap Operator (SO)* yaitu sebagai berikut:

1) $J_1 - J_3 - J_2 - J_4$ dengan *makespan* 543 menit
2) $J_4 - J_2 - J_3 - J_1$ dengan *makespan* 561 menit
3) $J_1 - J_2 - J_4 - J_3$ dengan *makespan* 536 menit

Dari hasil diatas diperoleh nilai $x_{j\,max} = 561$ dan $x_{j\,min} = 536$. Kemudian melakukan inisialisasi solusi awal dengan menggunakan persamaan (2.1) sehingga diperoleh hasil:

$x_{ij} = x_{j\,min} + rand(0,1).(x_{j\,max} - x_{j\,min})$

$x_{11} = 536 + (0,5)(561 - 536) = 548,5$

$x_{12} = 536 + (0,75)(561 - 536) = 554,75$

$x_{13} = 536 + (0,83)(561 - 536) = 556,75$

b. Menentukan Solusi Alternatif

Setelah penginisialisasian selesai maka pada tahap ini nilai yang dihasilkan akan diperluas dengan menggunakan persamaan (2.2).

$v_{ij} = x_{ij} + \emptyset_{ij}(x_{kj} - x_{ij})$

$v_{11} = 548,5 + (0,5)(554,75 - 548,5) = 551,625$

$v_{12} = 554,75 + (-0,7)(556,75 - 554,75) = 553,35$

$v_{13} = 556,75 + (-0,9)(548,5 - 556,75) = 564,175$

Kemudian dilakukan seleksi *greedy* yaitu dengan membandingkan nilai x_{ij} dan nilai v_{ij}. Jika nilai v_{ij} lebih kecil dari nilai x_{ij} maka nilai v_{ij} tersebut akan dianggap sama dengan nilai x_{ij} dan nilai percobaan tetap bernilai 0. Jika tidak, nilai x_{ij} yang disimpan dan nilai percobaan ke-*i* ditambah dengan 1. Hasil seleksi *greedy* sebagai berikut:

$$f(x_i) = \begin{cases} 548,5 & \text{dengan nilai percobaan 1} \\ 553,35 & \text{dengan nilai percobaan 0} \\ 556,75 & \text{dengan nilai percobaan 1} \end{cases}$$

c. Evaluasi Populasi Awal

Pada tahap ini akan dihitung nilai *fitness* pada setiap populasi dengan menggunakan persamaan (2.3).

$$fitness\ (x_1) = \frac{1}{(1 + 548,5)} = 0,00182$$

$$fitness\ (x_1) = \frac{1}{(1 + 553,35)} = 0,0018$$

$$fitness\ (x_1) = \frac{1}{(1 + 556{,}75)} = 0{,}00179$$

d. Evaluasi Populasi Alternatif

Setelah semua nilai *fitness* didapatkan maka akan dihitung nilai *probability* untuk semua kemungkinan solusi dengan menggunakan persamaan (2.4) yaitu sebagai berikut:

$$\sum_{i=1}^{SN} fitness_i = 0{,}00182 + 0{,}0018 + 0{,}00179 = 0{,}00541$$

$$p_1 = \frac{0{,}00182}{0{,}00541} = 0{,}33641$$

$$p_2 = \frac{0{,}0018}{0{,}00541} = 0{,}33272$$

$$p_3 = \frac{0{,}00179}{0{,}00541} = 0{,}33087$$

e. Evaluasi Populasi Akhir

Setelah didapatkan nilai *probability*, akan dibangkitkan nilai *random* antara 0 sampai 1. Jika nilai p_i lebih besar dari bilangan *random* yang ditentukan, maka akan memperluas kembali kemungkinan solusi yang terpilih menggunakan persamaan (2.2). Misalkan dibangkitkan nilai *random* yaitu 0,33115 maka untuk kemungkinan solusi p_1 dan p_2 akan diperluas kembali karena nilai p_1 dan p_2 lebih besar dari nilai *random* yang dibangkitkan. Sehingga kemungkinan solusi untuk keduanya di perluas sebagai berikut:

$v_{11} = 548{,}5 + (0{,}1)(554{,}75 - 548{,}5) = 549{,}125$

$v_{12} = 554{,}75 + (-0{,}8)(556{,}75 - 554{,}75) = 553{,}15$

Kemudian dilakukan seleksi *greedy* antara nilai x_{ij} dengan nilai baru v_{ij}. Sehingga diperoleh nilai sebagai berikut:

$$f(x_i) = \begin{cases} 548{,}5 & \text{dengan nilai percobaan 2} \\ 553{,}15 & \text{dengan nilai percobaan 0} \\ 556{,}75 & \text{dengan nilai percobaan 1} \end{cases}$$

Setelah diperoleh nilai percobaan yang maksimum, maka nilai kemungkinan solusi dengan nilai percobaan maksimum tersebut yang akan disimpan sebagai salah satu pilihan solusi terbaik dalam *list* solusi. Pada proses iterasi pertama ini diperoleh bahwa kemungkinan solusi pertama memiliki nilai percobaan maksimum yaitu 2, sehingga kemungkinan solusi tersebut akan disimpan. Pada iterasi berikutnya, solusi yang didapat akan dibandingkan kembali dengan solusi yang ada dalam *list.* Apabila solusi baru memiliki nilai yang lebih baik maka akan menggantikan solusi yang lama. Sehingga hasil optimal yang didapat pada iterasi pertama yaitu urutan jadwal $J_1 - J_3 - J_2 - J_4$ dengan *makespan* 543 menit.

Untuk iterasi selanjutnya kembali pada langkah b. Perhitungan ini dilakukan hingga iterasi yang ditentukan terpenuhi dan akan diperoleh urutan *job* yang optimal dengan nilai *makespan* minimum.

4.1.2 Penyelesaian Manual dengan Algoritma *Differential Evolution Plus*

Berikut ini penyelesaian penjadwalan *flowshop* secara manual dalam satu iterasi menggunakan algoritma *Differential Evolution Plus*. Namun sebelumnya di tentukan parameter yang digunakan terlebih dahulu misalnya populasi yang dibangkitkan = 3, batas atas (*Ub*) = 1, batas bawah (*Lb*) = 0, nilai terkecil probabilitas *crossover* $(Cr_{min}) = 0{,}2$, nilai terbesar probabilitas *crossover* $(Cr_{max}) = 0{,}7$, nilai minimum parameter mutasi $(F_{min}) = 0{,}5$, jumlah iterasi maksimum (*MAXGEN*) = 100. Langkah penyelesaiannya adalah sebagai berikut:

a. Inisialisasi Populasi

Pada tahap ini dibangkitkan sejumlah 3 populasi, dimana tiap populasi merepresentasikan sejumlah *job* dengan menentukan misal batas atas (*Ub*) sebesar 1 dan batas bawah (*Lb*) sebesar 0. Kemudian dihitung nilai individu pada tiap populasi (x_i) yang dibangkitkan menggunakan persamaan (2.5).

$$x_i = \begin{bmatrix} 0{,}251 & 0{,}153 & 0{,}415 & 0{,}512 \\ 0{,}679 & 0{,}739 & 0{,}151 & 0{,}201 \\ 0{,}257 & 0{,}891 & 0{,}993 & 0{,}154 \end{bmatrix}$$

Nilai x_i memiliki indeks sebagai berikut:

$$x_i = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

Untuk mendapatkan solusi awal, nilai individu pada tiap populasi diurutkan menggunakan prosedur SPV. Hasil pengurutannya adalah:

$$x_i = \begin{bmatrix} 0{,}153 & 0{,}251 & 0{,}415 & 0{,}512 \\ 0{,}151 & 0{,}201 & 0{,}679 & 0{,}739 \\ 0{,}154 & 0{,}257 & 0{,}891 & 0{,}993 \end{bmatrix}$$

Sehingga solusi awal pemutasi *job* sebagai berikut:

$$x_i = \begin{bmatrix} 2 & 1 & 3 & 4 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix}$$

Selanjutnya nilai fungsi objektif (*makespan*) pada tiap populasi dihitung dan dievaluasi untuk menentukan f_{min} dan f_{max} yang merupakan nilai minimum dan maksimum fungsi objektif.

Untuk urutan *job* : [2 1 3 4] , *makespan* $f(x_i)$ = 549 menit

Untuk urutan *job* : [3 4 1 2] , *makespan* $f(x_i)$ = 570 menit

Untuk urutan *job* : [4 1 2 3] , *makespan* $f(x_i)$ = 560 menit

Sehingga diperoleh nilai f_{min} = 549 dan f_{max} = 570.

b. *Crossover*

Tahap *crossover* digunakan untuk menentukan populasi yang akan dimutasi atau tidak. Sebelumnya telah di tentukan nilai parameter Cr_{min}, Cr_{max}, dan *MAXGEN*. Kemudian dihitung nilai parameter *crossover* (*Cr*) menggunakan persamaan (2.6) sehingga diperoleh hasil:

$$Cr = Cr_{min} + G\frac{Cr_{max} - Cr_{min}}{MAXGEN}$$

$$= 0{,}2 + 1\left(\frac{0{,}7 - 0{,}2}{100}\right)$$

$$= 0{,}2 + 0{,}005$$

$$= 0{,}205$$

Dibangkitkan bilangan *random* pada tiap populasi, apabila bilangan *random* $\leq Cr$ maka individu pada populasi ke – i akan dimutasi, apabila bilangan *random* $> Cr$ maka individu pada populasi ke – i tidak dimutasi.

Misal bilangan *random* yang dibangkitkan adalah:

$x_1 = 0{,}115 < Cr$

$x_2 = 0{,}205 = Cr$

$x_3 = 0{,}793 > Cr$

Maka populasi yang akan dimutasi adalah individu pada populasi $1(x_1)$ dan populasi $2(x_2)$.

c. Mutasi

Sebelum dilakukan mutasi, akan dihitung nilai parameter mutasi (F) menggunakan persamaan (2.8).

$$\left|\frac{f_{max}}{f_{min}}\right| = \left|\frac{570}{549}\right| = 1{,}038$$

Karena nilai $\left|\frac{f_{max}}{f_{min}}\right| > 1\ maka$

$$\mathrm{F} = \max\left(F_{min}, 1 - \left|\frac{f_{min}}{f_{max}}\right|\right)$$

$$= \max\left(0{,}5\ , 1 - \left|\frac{549}{570}\right|\right)$$

$$= \max(0{,}5\ , (1 - 0{,}963))$$

$$= \max(0{,}5\ , 0{,}037)$$

$$= 0{,}5$$

Kemudian lakukan mutasi pada x_1 dan x_2 dengan persamaan (2.7) dimana $x_{r0,g}, x_{r1,g}, x_{r2,g}$ pada persamaan tersebut merupakan nilai vektor yang dipilih secara acak dari nilai x_i pada tahap inisialisasi populasi sehingga diperoleh hasil mutasi pada populasi ke-i (v_i) berikut ini:

$$v_i = \begin{bmatrix} 1{,}312 & 0{,}235 & -0{,}218 & 0{,}327 \\ 0{,}537 & 0{,}254 & 0{,}714 & -0{,}267 \\ 0{,}257 & 0{,}891 & 0{,}993 & 0{,}154 \end{bmatrix}$$

Cek hasil mutasi apakah diluar batas atas dan batas bawah. Jika hasil mutasi lebih dari batas atas maka akan diganti dengan 1, jika hasil mutasi kurang dari batas bawah maka akan diganti dengan 0, sedangkan hasil mutasi yang bernilai antara 0-1 maka nilainya tetap. Sehingga hasilnya adalah:

$$v_i = \begin{bmatrix} 1 & 0{,}235 & 0 & 0{,}327 \\ 0{,}537 & 0{,}254 & 0{,}714 & 0 \\ 0{,}257 & 0{,}891 & 0{,}993 & 0{,}154 \end{bmatrix}$$

Permutasi *job*: $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$

Urutkan hasil perhitungan v_i diatas menggunakan prosedur SPV.

$$v_i = \begin{bmatrix} 0 & 0{,}235 & 0{,}327 & 1 \\ 0 & 0{,}254 & 0{,}537 & 0{,}714 \\ 0{,}154 & 0{,}257 & 0{,}891 & 0{,}993 \end{bmatrix}$$

Permutasi *job*: $\begin{bmatrix} 3 & 2 & 4 & 1 \\ 4 & 2 & 1 & 3 \\ 4 & 1 & 2 & 3 \end{bmatrix}$

Berdasarkan urutan permutasi *job* yang telah diperoleh, maka dihitung nilai *makespan* untuk setiap populasi.

Untuk urutan *job* : $[3 \quad 2 \quad 4 \quad 1]$, *makespan* $f(v_i)$ = 573 menit

Untuk urutan *job* : $[4 \quad 2 \quad 1 \quad 3]$, *makespan* $f(v_i)$ = 560 menit

Untuk urutan *job* : $[4 \quad 1 \quad 2 \quad 3]$, *makespan* $f(v_i)$ = 560 menit

d. Seleksi

Pada tahap ini akan dibandingkan nilai *makespan* hasil mutasi $f(v_i)$ dan nilai *makespan* awal $f(x_i)$ sehingga diperoleh nilai *makespan* yang minimum untuk tiap populasi ke-i $(f(x_i))$. Jika $f(v_i) < f(x_i)$ maka $x_i = v_i$, jika tidak maka $x_i = x_i$. Hasil seleksi sebagai berikut:

Dengan nilai $x_i = \begin{bmatrix} 0{,}153 & 0{,}253 & 0{,}415 & 0{,}512 \\ 0 & 0{,}254 & 0{,}537 & 0{,}714 \\ 0{,}154 & 0{,}257 & 0{,}891 & 0{,}993 \end{bmatrix}$

Permutasi *job*: $\begin{bmatrix} 2 & 1 & 3 & 4 \\ 4 & 2 & 1 & 3 \\ 4 & 1 & 2 & 3 \end{bmatrix}$

Untuk urutan *job* : [2 1 3 4] , *makespan* $f(x_i)$ = 549 menit

Untuk urutan *job* : [4 2 1 3] , *makespan* $f(x_i)$ = 560 menit

Untuk urutan *job* : [4 1 2 3] , *makespan* $f(x_i)$ = 560 menit

e. *Local Search*

Langkah *local search* ini dilakukan menggunakan prosedur *insert-based local search* yaitu prosedur dengan menyisipkan *job* ke-*u* pada posisi *v* dimana $u \neq v$. Pada tahap seleksi, telah diperoleh urutan permutasi *job* beserta nilai objektif, kemudian hasil seleksi tersebut disebut $f(\pi_{i\_0})$ yang merupakan nilai *makespan* permutasi *job* untuk populasi ke–*i* tahap *local search* ke–0. Pilih secara acak *job* ke-*u* dan posisi *v* dimana $u \neq v$, selanjutnya sisipkan *job u* ke posisi *v* pada setiap populasi serta hitung nilai *makespan* yang disebut dengan $f(\pi_i)$ yaitu nilai *makespan* permutasi *job* untuk populasi ke-*i*. Berikut ini hasil dari $f(\pi_i)$:

Untuk urutan *job* : [3 1 2 4] , *makespan* $f(\pi_i)$ = 569 menit

Untuk urutan *job* : [2 4 1 3] , *makespan* $f(\pi_i)$ = 542 menit

Untuk urutan *job* : [4 1 3 2] , *makespan* $f(\pi_i)$ = 568 menit

Dalam *insert-based local search*, proses penyisipan *job u* ke posisi *v* dilakukan *looping* beberapa kali. Untuk *looping* pertama diperoleh:

Untuk urutan *job* : [2 1 3 4] , *makespan* $f(\pi_{i\_1})$ = 549 menit

Untuk urutan *job* : [2 1 4 3] , *makespan* $f(\pi_{i\_1})$ = 542 menit

Untuk urutan *job* : [4 3 1 2] , *makespan* $f(\pi_{i\_1})$ = 568 menit

Bandingkan $f(\pi_i)$ dan $f(\pi_{i\_1})$, urutan *job* dengan nilai *makespan* lebih baik akan terpilih sehingga diperoleh hasil perbandingan:

Untuk urutan *job* : [2 1 3 4] , *makespan* $f(\pi_i)$ = 549 menit

Untuk urutan *job* : [2 4 1 3] , *makespan* $f(\pi_i)$ = 542 menit

Untuk urutan *job* : [4 1 3 2] , *makespan* $f(\pi_i)$ = 568 menit

Lakukan langkah seperti diatas hingga kriteria pemberhentian *looping* yaitu $< n(n-1)$. Setelah *looping* dilakukan hingga kriteria pemberhentian, maka diperoleh urutan *job* beserta nilai *makespan* yang disebut $f(\pi_i)$ dari hasil *looping*.

Untuk urutan *job* : [1 2 4 3] , *makespan* $f(\pi_i)$ = 536 menit

Untuk urutan *job* : [2 4 1 3] , *makespan* $f(\pi_i)$ = 542 menit

Untuk urutan *job* : [2 3 4 1] , *makespan* $f(\pi_i)$ = 543 menit

Bandingkan $f(\pi_i)$ dari hasil *looping* dengan $f(\pi_{i\_0})$ pada tahap seleksi sehingga akan diperoleh urutan *job*, nilai *makespan* dan nilai x_i. Hasilnya sebagai berikut:

Untuk urutan *job* : [1 2 4 3] , *makespan* $f(x_i)$ = 536 menit

Untuk urutan *job* : [2 4 1 3] , *makespan* $f(x_i)$ = 542 menit

Untuk urutan *job* : [2 3 4 1] , *makespan* $f(x_i)$ = 543 menit

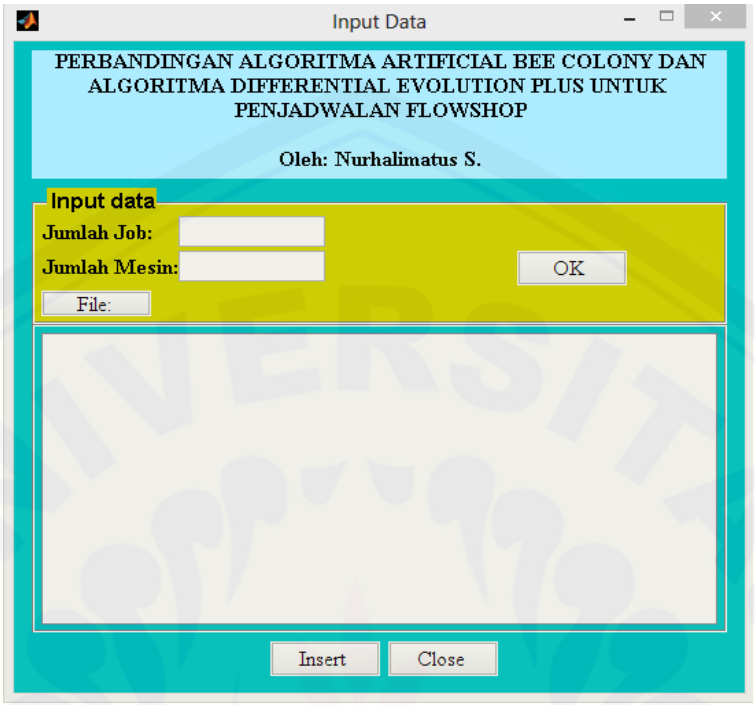
$$x_i = \begin{bmatrix} 0{,}253 & 0{,}153 & 0{,}512 & 0{,}415 \\ 0{,}254 & 0 & 0{,}537 & 0{,}714 \\ 0{,}891 & 0{,}993 & 0{,}154 & 0{,}257 \end{bmatrix}$$

Dari perhitungan yang telah dilakukan, maka diperoleh hasil optimum pada iterasi pertama yaitu urutan jadwal $J_1 - J_2 - J_4 - J_3$ dengan *makespan* 536 menit.

Untuk iterasi selanjutnya, kembali pada proses langkah b. Proses dalam algoritma *Differential Evolution Plus* ini dilakukan hingga iterasi maksimal terpenuhi dan akan diperoleh urutan *job* yang optimal dengan nilai *makespan* minimum.

4.2 Penyelesaian Penjadwalan *Flowshop* Menggunakan Program

Pada skripsi ini, disertakan program penjadwalan *flowshop* menggunakan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* yang dibuat dengan bantuan *software* Matlab. Program ini dibuat untuk mempercepat dan mempermudah dalam melakukan perhitungan kedua algoritma. Tampilan awal dari program yang telah dibuat ditunjukkan pada Gambar 4.1.



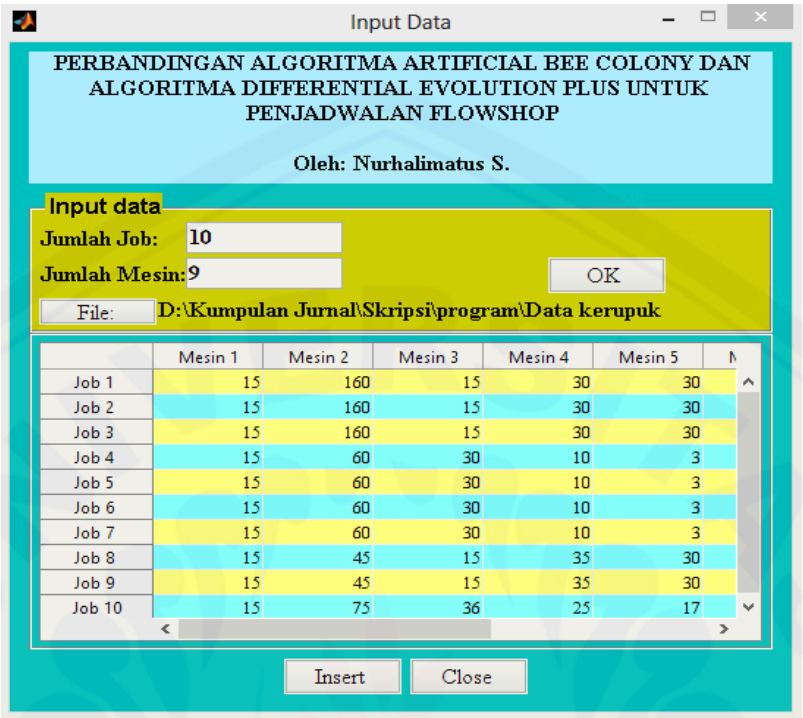
Gambar 4.1 Tampilan awal program aplikasi

Terdapat beberapa langkah yang harus dilakukan untuk menjalankan program aplikasi penjadwalan *flowshop*, berikut penjelasan setiap langkah-langkah untuk menjalakan program aplikasi penjadwalan *flowshop.*

a. Langkah awal untuk menjalankan pogram yaitu meng-*input* data yang akan digunakan. Terdapat dua cara untuk meng-*input* data yaitu pertama bisa dengan cara memasukkan jumlah *job* dan jumlah mesin yang akan diproses pada kolom "*Input* data". Pada skripsi ini menggunakan jumlah *job* sebanyak sepuluh *job* dan jumlah mesin sebanyak sembilan mesin. Kemudian klik tombol "OK". Sehingga akan muncul tabel untuk meng-*input* data, seperti pada Gambar 4.2. Setelah itu *input* data waktu proses tiap *job* pada masing-masing mesin sesuai data yang ada pada Tabel 3.1.

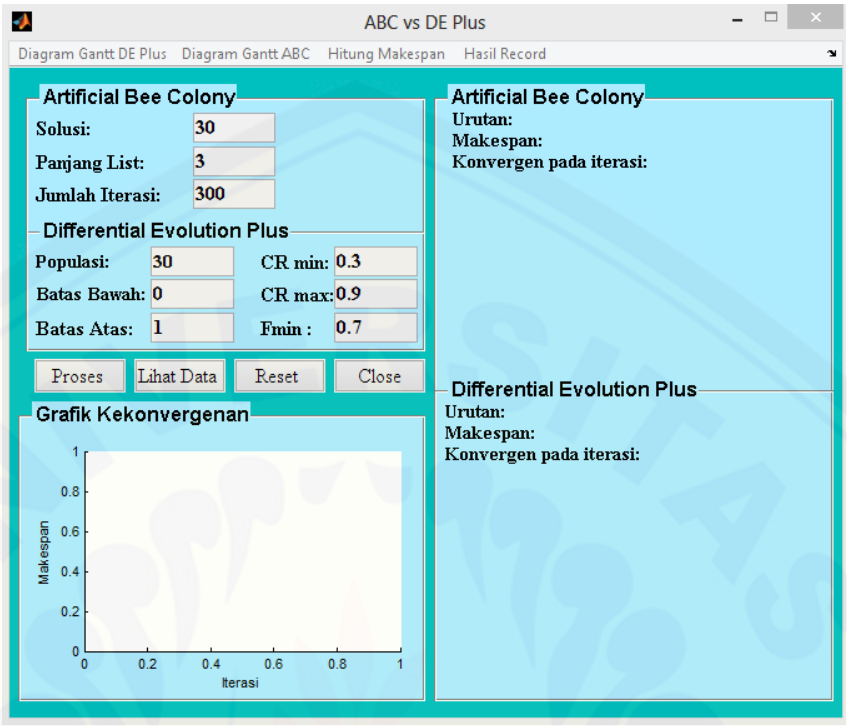
Gambar 4.2 Tampilan *input* data

Cara yang kedua yaitu langsung membuka file data waktu proses pembuatan kerupuk dengan cara klik tombol “File” dan pilih file “Data kerupuk samjaya.txt” maka akan muncul data seperti pada Gambar 4.3. Kemudian untuk proses selanjutnya klik tombol “insert”.



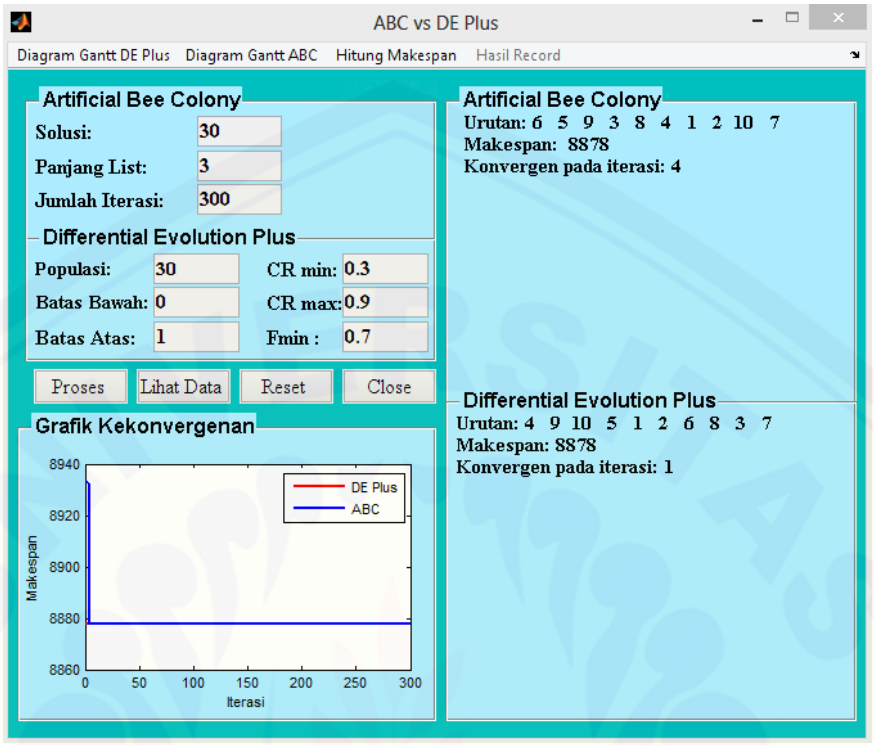
Gambar 4.3 Tampilan *input* data waktu pembuatan kerupuk

b. Setelah itu akan mucul tampilan seperti yang ada pada Gambar 4.4. Pada gambar tersebut harus meng-*input* nilai parameter awal pada masing-masing algoritma sebelum memproses data. Pada algoritma *Artificial Bee Colony* parameter awal yang digunakan terdiri dari solusi = 25, panjang *list* = 3. Sedangkan parameter awal yang digunakan pada algoritma *Differential Evolution Plus* terdiri dari populasi = 20, batas bawah = 0,5, batas atas = 0,6, $Cr_{min} = 0{,}5$, $Cr_{max} = 0{,}9$, dan $F_{min} = 0{,}8$. Kemudian masukkan nilai iterasi yaitu 200. Kemudian pilih menu "Proses" untuk mengetahui hasil dari kedua algoritma. Selain itu juga terdapat menu "Lihat Data" digunakan untuk menampilkan data waktu proses pada setiap *job* pada setisp mesin yang telah di *input*kan sebelumnya. Menu "Reset" digunakan untuk mengulangi semua proses dan menu "Close" untuk menutup program.



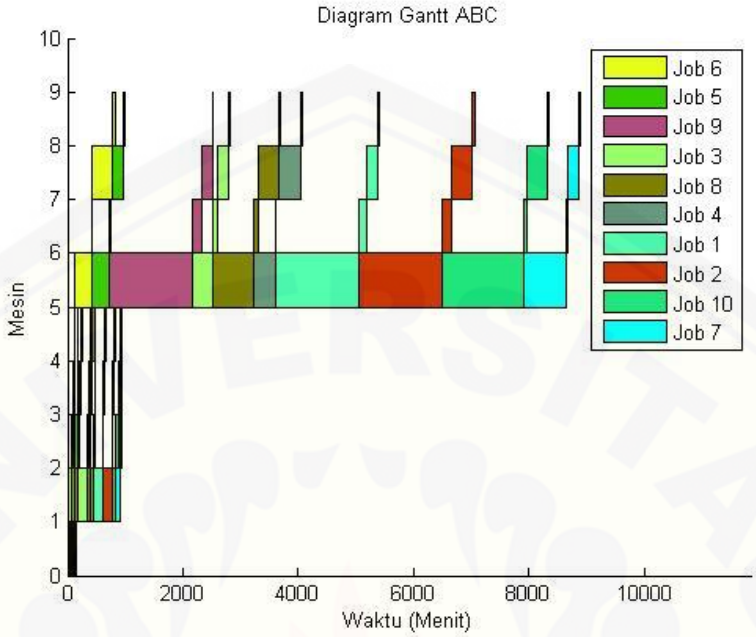
Gambar 4.4 Tampilan proses data

c. Hasil akhir yang ditampilkan dalam perhitungan menggunakan program ini adalah urutan penjadwalan *job*, nilai *makespan*, kekonvergenan iterasi, grafik kekonvergenan, dan gambar diagram *gantt* yang dihasilkan oleh masing-masing algoritma. Pada Gambar 4.5 menunjukkan hasil optimal dari algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* dalam sekali pengujian dengan nilai parameter awal yang telah di *input*.

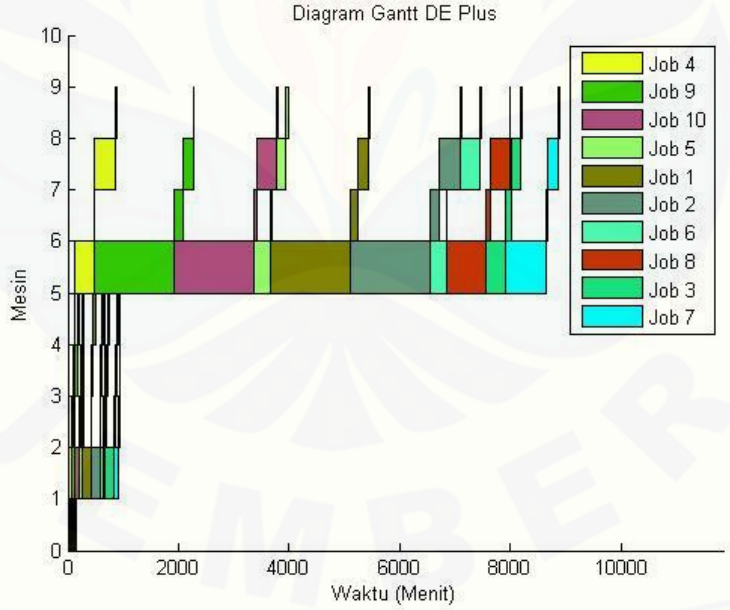


Gambar 4.5 Tampilan *output*

Berdasarkan Gambar 4.5 dapat diketahui bahwa hasil optimal dari algoritma *Artificial Bee Colony* diperoleh urutan jadwal $J_6 - J_5 - J_9 - J_3 - J_8 - J_4 - J_1 - J_2 - J_{10} - J_7$ dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-4. Sedangkan hasil optimal dari algoritma *Differential Evolution Plus* diperoleh urutan jadwal $J_4 - J_9 - J_{10} - J_5 - J_1 - J_2 - J_6 - J_8 - J_3 - J_7$ dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1. Sedangkan Diagram *gantt* yang dihasilkan dari kedua algoritma tersebut dapat dilihat pada Gambar 4.6 (a) dan Gambar 4.6 (b). Diagram *gantt* tersebut merupakan hasil akhir perhitungan pada kedua algoritma. Urutan mesin dari masing-masing algoritma digambarkan dengan sumbu horisontal dan sumbu vertikal menggambarkan pergerakan waktu. Perbedaan warna yang digunakan pada *gantt chart* bertujuan untuk memudahkan pengguna dalam membedakan waktu yang diperlukan oleh tiap *job* ketika diproses dalam urutan mesin yang ada.



(a)



(b)

Gambar 4.6 Tampilan diagram *gantt* (a) Algoritma ABC; (b) Algoritma DE Plus

4.3 Pebandingan Algoritma *Artificial Bee Colony* dan Algoritma *Differential Evolution Plus*

Pengujian algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* yang diterapkan pada data penjadwalan produksi kerupuk UD. Samjaya dilakukan sembilan kali percobaan dengan nilai parameter yang berbeda. Masing-masing percobaan dilakukan 10 kali *running*, dengan tujuan untuk mendapatkan hasil terbaik yang akan dipilih. Berikut ini adalah hasil percobaan yang dilakukan.

a. Percobaan pertama parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 5, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 5, Lb = 0, Ub = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 100. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil percobaan pertama

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	47	8878	12
2	8878	39	8878	4
3	8878	9	8878	6
4	8878	39	8878	7
5	8878	64	8878	5
6	8878	48	8878	10
7	8878	40	8878	3
8	8878	32	8878	3
9	8900	94	8878	3
10	8878	38	8878	4

Berdasarkan Tabel 4.3 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-9. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-3.

b. Percobaan kedua parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 10, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 10, Lb = 0, Ub = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 100. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil percobaan kedua

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	12	8878	5
2	8878	11	8878	4
3	8878	51	8878	3
4	8878	29	8878	3
5	8878	17	8878	5
6	8878	30	8878	4
7	8878	8	8878	4
8	8878	11	8878	6
9	8878	31	8878	3
10	8878	15	8878	3

Berdasarkan Tabel 4.4 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-8. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-3.

c. Percobaan ketiga parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 30, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 30, Lb = 0, Ub = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 100. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil percobaan ketiga

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	9	8878	2
2	8878	31	8878	1
3	8878	37	8878	1
4	8878	32	8878	1
5	8878	2	8878	1
6	8878	21	8878	1
7	8878	14	8878	1
8	8878	5	8878	1
9	8878	11	8878	1
10	8878	7	8878	1

Berdasarkan Tabel 4.5 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan

konvergen pada iterasi ke-2. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1.

d. Percobaan keempat parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 5, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 5, *Lb* = 0, *Ub* = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 200. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil percobaan keempat

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	48	8878	7
2	8878	100	8878	3
3	8878	27	8878	4
4	8878	12	8878	3
5	8878	15	8878	4
6	8878	23	8878	2
7	8878	26	8878	4
8	8878	13	8878	4
9	8878	8	8878	5
10	8878	42	8878	5

Berdasarkan Tabel 4.6 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-8. Sedangkan pada algoritma *Differential Evolution*

Plus mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-2.

e. Percobaan kelima parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 10, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 10, Lb = 0, Ub = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 200. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.7.

Tabel 4.7 Hasil percobaan kelima

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	33	8878	3
2	8878	12	8878	4
3	8878	4	8878	2
4	8878	14	8878	7
5	8878	11	8878	3
6	8878	4	8878	5
7	8878	85	8878	4
8	8878	5	8878	3
9	8878	10	8878	2
10	8878	20	8878	4

Berdasarkan Tabel 4.7 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-4. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-2.

f. Percobaan keenam parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 30, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 30, *Lb* = 0, *Ub* = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 200. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.8.

Tabel 4.8 Hasil percobaan keenam

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	3	8878	1
2	8878	1	8878	1
3	8878	49	8878	1
4	8878	27	8878	1
5	8878	1	8878	1
6	8878	10	8878	1
7	8878	2	8878	1
8	8878	21	8878	1
9	8878	12	8878	1
10	8878	15	8878	1

Berdasarkan Tabel 4.8 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1.

g. Percobaan ketujuh parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 5, panjang *list* solusi = 3. Sedangkan parameter pada

algoritma *Differential Evolution Plus* yaitu populasi = 5, Lb = 0, Ub = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 300. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.9.

Tabel 4.9 Hasil percobaan ketujuh

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	24	8878	4
2	8878	17	8878	2
3	8878	42	8878	7
4	8878	34	8878	2
5	8878	14	8878	4
6	8878	4	8878	3
7	8878	48	8878	4
8	8878	21	8878	2
9	8878	8	8878	9
10	8878	22	8878	3

Berdasarkan Tabel 4.9 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-4. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-2.

h. Percobaan kedelapan parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 10, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 10, Lb = 0, Ub = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua

algoritma adalah 300. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.10.

Tabel 4.10 Hasil percobaan kedelapan

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	9	8878	5
2	8878	12	8878	3
3	8878	21	8878	3
4	8878	2	8878	3
5	8878	32	8878	9
6	8878	4	8878	5
7	8878	60	8878	4
8	8878	7	8878	1
9	8878	9	8878	2
10	8878	2	8878	1

Berdasarkan Tabel 4.10 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-2. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1.

i. Percobaan kesembilan parameter yang digunakan pada algoritma *Artificial Bee Colony* yaitu solusi = 30, panjang *list* solusi = 3. Sedangkan parameter pada algoritma *Differential Evolution Plus* yaitu populasi = 30, *Lb* = 0, *Ub* = 1, $Cr_{min} = 0{,}3$, $Cr_{max} = 0{,}9$, $F_{min} = 0{,}7$. Jumlah iterasi yang digunakan kedua algoritma adalah 300. Kemudian dengan parameter tersebut dilakukan 10 kali *running* dan hasilnya dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil percobaan kesembilan

Running ke-	Algoritma *Artificial Bee Colony*		Algoritma *Differential Evolution Plus*	
	Makespan	Iterasi kekonvergenan	*Makespan*	Iterasi kekonvergenan
1	8878	2	8878	1
2	8878	23	8878	1
3	8878	31	8878	1
4	8878	4	8878	1
5	8878	3	8878	1
6	8878	1	8878	1
7	8878	14	8878	1
8	8878	5	8878	1
9	8878	6	8878	1
10	8878	1	8878	1

Berdasarkan Tabel 4.11 dapat dilihat bahwa pada algoritma *Artificial Bee Colony* mencapai hasil yang optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1. Sedangkan pada algoritma *Differential Evolution Plus* mencapai hasil optimal dengan nilai *makespan* yaitu 8.878 menit dan konvergen pada iterasi ke-1.

Rangkuman setelah dilakukan sembilan kali percobaan dengan menggunakan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* dapat dilihat pada Tabel 4.12 dan Tabel 4.13. Kedua tabel tersebut diperoleh dari hasil optimal dengan konvergensi terbaik pada setiap percobaan yang telah dilakukan.

Tabel 4.12 Rangkuman sembilan percobaan dengan algoritma *Artificial Bee Colony*

Percobaan ke-	Urutan Jadwal	*Makespan* (menit)	Iterasi Konvergen
1	$J_4 - J_{10} - J_5 - J_3 - J_9 - J_1 - J_2 - J_6 - J_8 - J_7$	8878	9
2	$J_6 - J_{10} - J_8 - J_4 - J_2 - J_5 - J_3 - J_1 - J_9 - J_7$	8878	8
3	$J_5 - J_3 - J_1 - J_6 - J_2 - J_8 - J_4 - J_{10} - J_9 - J_7$	8878	2
4	$J_6 - J_3 - J_4 - J_{10} - J_1 - J_8 - J_9 - J_2 - J_7 - J_5$	8878	8
5	$J_5 - J_4 - J_9 - J_{10} - J_8 - J_2 - J_6 - J_1 - J_3 - J_7$	8878	4
6	$J_5 - J_2 - J_4 - J_{10} - J_9 - J_6 - J_3 - J_1 - J_8 - J_7$	8878	1
7	$J_5 - J_9 - J_3 - J_2 - J_1 - J_4 - J_8 - J_{10} - J_6 - J_7$	8878	4
8	$J_6 - J_4 - J_8 - J_2 - J_{10} - J_7 - J_9 - J_3 - J_1 - J_5$	8878	2
9	$J_5 - J_6 - J_8 - J_{10} - J_1 - J_3 - J_9 - J_2 - J_4 - J_7$	8878	1

Tabel 4.13 Rangkuman sembilan percobaan dengan algoritma *Differential Evolution Plus*

Percobaan ke-	Urutan Jadwal	*Makespan* (menit)	Iterasi Konvergen
1	$J_4 - J_3 - J_{10} - J_6 - J_9 - J_5 - J_8 - J_1 - J_2 - J_7$	8878	3
2	$J_5 - J_{10} - J_9 - J_2 - J_3 - J_1 - J_8 - J_4 - J_6 - J_7$	8878	3
3	$J_4 - J_{10} - J_6 - J_8 - J_9 - J_2 - J_3 - J_5 - J_1 - J_7$	8878	1
4	$J_4 - J_9 - J_8 - J_3 - J_{10} - J_6 - J_5 - J_2 - J_1 - J_7$	8878	2
5	$J_4 - J_5 - J_{10} - J_9 - J_3 - J_8 - J_1 - J_2 - J_6 - J_7$	8878	2
6	$J_4 - J_2 - J_8 - J_3 - J_7 - J_6 - J_{10} - J_9 - J_1 - J_5$	8878	1
7	$J_5 - J_3 - J_{10} - J_9 - J_8 - J_1 - J_2 - J_6 - J_4 - J_7$	8878	2
8	$J_5 - J_3 - J_9 - J_6 - J_8 - J_{10} - J_4 - J_1 - J_2 - J_7$	8878	1
9	$J_4 - J_8 - J_2 - J_{10} - J_5 - J_9 - J_1 - J_3 - J_6 - J_7$	8878	1

Berdasarkan hasil percobaan pada algoritma *Artificial Bee Colony* yang telah dilakukan dengan menambahkan jumlah solusi dan jumlah iterasi, terlihat bahwa kedua parameter tersebut memberikan efek yang signifikan terhadap performa algoritma *Artificial Bee Colony*. Semakin besar jumlah solusi dan jumlah iterasinya maka dapat menghasilkan nilai *makespan* yang selalu optimal dan cepat konvergen. Sedangkan pada algoritma *Differential Evolution Plus* terlihat bahwa dengan melakukan penambahan jumlah populasi dan jumlah iterasi dapat berpengaruh terhadap kinerja algoritma *Differential Evolution Plus*. Semakin besar jumlah populasi dan jumlah iterasinya maka dapat konsisten dalam menghasilkan nilai *makespan* yang selalu optimal dan cepat konvergen.

Pada percobaan yang telah dilakukan dengan algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* menghasilkan nilai *makespan* yang sama yaitu 8.878 menit, tetapi urutan jadwal yang dihasilkan berbeda pada setiap percobaannya. Hal ini terjadi karena pada saat proses terdapat bilangan *random* yang dibangkitkan. Parameter-parameter yang digunakan pada setiap algoritma sudah mewakili selang yang telah ditentukan.

Dari percobaan tersebut, diperoleh salah satu urutan jadwal dari algoritma *Artificial Bee Colony* yaitu $J_5 - J_6 - J_8 - J_{10} - J_1 - J_3 - J_9 - J_2 - J_4 - J_7$ dengan nilai *makespan* yaitu 8.878 menit. Sedangkan salah satu urutan jadwal yang diperoleh dari algoritma *Differential Evolution Plus* yaitu $J_4 - J_8 - J_2 - J_{10} - J_5 - J_9 - J_1 - J_3 - J_6 - J_7$ dengan nilai *makespan* yaitu 8.878 menit. Keadaan tersebut menunjukkan bahwa kedua algoritma tersebut memiliki tingkat efektifitas yang sama jika diterapkan pada penjadwalan produksi kerupuk UD. Samjaya. Apabila ditinjau dari tingkat kekonvergensinya, maka menunjukkan bahwa algoritma *Differential Evolution Plus* lebih cepat konvergen dari pada algoritma *Artificial Bee Colony*.

BAB 5. PENUTUP

5.1 Kesimpulan

Dari penelitian yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

a. Penjadwalan *flowshop* yang diterapkan pada produksi kerupuk UD. Samjaya dengan menggunakan algoritma *Artificial Bee Colony* menghasilkan urutan jadwal yaitu kerupuk mawar 2 udang kecil – kerupuk mawar SI besar – kerupuk kasandra besar – kerupuk obar-abir – kerupuk impala panjang – kerupuk Rajang – kerupuk kasandra kecil – kerupuk impala pendek – kerupuk mawar 2 udang besar – kerupuk mawar SI kecil dengan nilai *makespan* 8.878 menit. Sedangkan pada algoritma *Differential Evolution Plus* menghasilkan urutan jadwal yaitu kerupuk mawar 2 udang besar – kerupuk kasandra besar – kerupuk impala pendek – kerupuk obar-abir – kerupuk mawar 2 udang kecil – kerupuk kasandra kecil – kerupuk impala panjang – kerupuk rajang – kerupuk mawar SI besar – kerupuk mawar SI kecil dengan nilai *makespan* 8.878 menit. Meskipun dilakukan sembilan kali percobaan, nilai *makespan* yang dihasilkan tetap sama yaitu 8.878 menit. Akan tetapi urutan jadwal yang dihasilkan pada setiap pengujian adalah berbeda. Hal tersebut menunjukkan bahwa kedua algoritma tersebut memiliki efektifitas yang sama untuk diterapkan pada penjadwalan *flowshop* ini.
b. Algoritma *Differential Evolution Plus* memiliki tingkat kecepatan kekonvergenan yang lebih baik dibandingkan dengan algoritma *Artificial Bee Colony* karena lebih cepat konvergen.

5.2 Saran

Algoritma *Artificial Bee Colony* dan algoritma *Differential Evolution Plus* merupakan algoritma yang digunakan untuk menyelesaikan permasalahan optimasi sehingga bisa diterapkan pada permasalahan lain selain *flowshop* yaitu seperti *jobshop*, TSP, *knapsack* dan lain sebagainya. Selain itu, juga dapat dikembangkan dengan membandingkannya menggunakan metode yang lain yaitu seperti cat swarm optimization, ant colony optimization, dan lain sebagainya.

DAFTAR PUSTAKA

Baker, K. 1974. *Introduction to Sequencing and Scheduling*. New York: John Wiley and Sons,Inc.

Chong, C.S., Low, M.Y.H., Sivakumar, A.I., Gay, K.L. 2006. *A Bee Colony Optimization Algorithm to Job Shop Scheduling*. Singapore: Nayang Technological University.

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

Kuncoro, C. 2013. "Penjadwalan Produksi Kertas Menggunakan Algoritma Pour dan Algoritma NEH di PT. Kertas Leces Probolinggo". Tidak Diterbitkan. Skripsi. Jember: Universitas Jember.

Nugraha, D.W. 2012. Penerapa Kompleksitas Waktu Algoritma Prim Untuk Menghitung Kemampuan Komputer Dalam Melaksanakan Perintah. *Jurnal Ilmiah Foristek*. **2**: 195-207.

Nurjanah, W.D. 2015. "Penerapan Algoritma Genetika dan algoritma Simulated Anneling pada Penjadwalan Flowshop". Tidak diterbitkan. Skripsi. Jember: FMIPA Universitas Jember.

Santosa, B., dan Willy, P. 2011. *Metoda Metaheuristik: Konsep dan Impelementasi*. Surabaya: Guna Widya.

Sari, T.K. 2015. "Perbandingan Algoritma Simulated Annealing dan algoritma Differential Evolution Plus Untuk Penjadwalan Flowshop". Tidak diterbitkan. Skripsi. Jember: FMIPA Universitas Jember.

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

Sugioko, A. 2013. "Perbandingan Algoritma Bee Colony dengan Algoritma Bee Colony Tabu List dalam Penjadwalan Flowshop". *Jurnal Metris*. **14**: 113-120.

Suyanto. 2010. *Algoritma Optimasi.* Yogyakarta: Graha Ilmu.

Wiratno, S.E., Nurdiansyah, R., dan Santosa, B. 2012. "Algoritma Differential Evolution Untuk Penjadwalan Flowshop Banyak Mesin Dengan Multi Obyektif". *Jurnal Teknik Industri Institut Teknologi Sepuluh Nopember*. **13**: 1-6.