ISSN 2089-6131 (Print)
ISSN 2443-1311 (Online)
DOI 10.22146/teknosains.22901
https://jurnal.ugm.ac.id/teknosains

JURNAL TEKNOSAINS

VOLUME 6 No. 2, 22 Juni 2017 Halaman 59-138

PENYELESAIAN *MULTI-OBJECTIVE FLEXIBLE JOB SHOP SCHEDULING PROBLEM* MENGGUNAKAN *HYBRID* ALGORITMA IMUN

Yabunayya Habibi, Galandaru Swalaganata*, dan *Aprilia Divi Yustita
Fakultas Matematika dan Ilmu Pengetahuan Alam
Institut Teknologi Sepuluh Nopember
Email: habibiexebppt@gmail.com

ABSTRACT

Flexible Job shop scheduling problem (FJSSP) is one of scheduling problems with specification: there is a job to be done in a certain order, each job contains a number of operations and each operation is processed on a machine of some available machine. The purpose of this paper is to solve Multi-objective Flexible Job Shop scheduling problem with minimizing the makespan, the biggest workload and the total workload of all machines. Because of complexity these problem, a integrated approach Immune Algorithm (IA) and Simulated Annealing (SA) algorithm are combined to solve the multi-objective FJSSP. A clonal selection is a strategy for generating new antibody based on selecting the antibody for reproduction. SA is used as a local search search algorithm for enhancing the local ability with certain probability to avoid becoming trapped in a local optimum. The simulation result have proved that this hybrid immune algorithm is an efficient and effective approach to solve the multi-objective FJSSP.

Keywords: ***Flexible Job Shop Scheduling; Immune Algorithm; Multi-objective optimization; Simulated Annealing.***

ABSTRAK

Flexible Job shop scheduling problem (FJSSP) merupakan salah satu masalah penjadwalan dengan spesifikasi: ada sejumlah *job* dengan urutan tertentu yang harus dikerjakan, setiap *job* memuat sejumlah operasi dan setiap operasi diproses pada satu mesin dari beberapa alternatif mesin yang tersedia. Tujuan penulisan ini adalah menyelesaikan *Multi-objective Flexible Job shop scheduling problem* dengan kriteria meminimumkan *makespan*, *workload* terbesar, dan total *workload* dari seluruh mesin.Untuk itu, kompleksitas permasalahan tersebut, sebuah pendekatan terintegrasi algoritma Imun dan *Simulated Annealing* dikombinasikan untuk menyelesaikan *multi-objective* FJSSP. Sebuah seleksi klonal merupakan suatu cara untuk pembentukan antibodi baru berbasis pada seleksi antibodi untuk reproduksi. SA digunakan sebagai sebuah algoritma pencarian lokal untuk meningkatkan kemampuan lokal dengan probabilitas tertentu untuk menghindari solusi optimal lokal. Hasil simulasi menunjukkan bahwa *hybrid* algoritma Imun merupakan pendekatan efektif dan efisien dalam menyelesaikan *multi-objective* FJSSP.

Kata Kunci: ***Algoritma Imun; Flexible Job Shop Scheduling Problem; Optimasi Multi-Objective; Simulated Annealing.***

PENGANTAR

Penjadwalan merupakan aktivitas yang penting untuk meningkatkan produktifitas dalam perencanaan dan pengelolaan (Chaudhry dkk, 2013). Permasalahan penjadwalan hampir ditemukan di mana-mana seperti pada masalah produksi, distribusi, manajemen, transportasi, manufaktur, dan lain-lain. Dalam bidang industri, perusahaan berupaya untuk memiliki penjadwalan yang baik dalam sistem produksi sehingga dapat meningkatkan hasil dengan biaya operasional yang minimal, proses produksi yang efisien dengan waktu seminimal mungkin. Sistem produksi yang melibatkan banyak proses, mesin, dan waktu proses yang bervariasi akan menemui banyak hambatan dan terganggunya proses produksi bila sumber daya tidak dikelola dengan tepat.

Job shop scheduling problem merupakan cabang dari penjadwalan produksi yang merupakan salah satu masalah optimasi kombinatorial yang sulit untuk diselesaikan (Baykasoglu, 2004). *Job Shop Scheduling Problem* (JSSP) secara umum merupakan masalah penjadwalan pada sejumlah *n-job* pada sejumlah *m*-mesin dengan tujuan untuk meminimalkan kriteria tertentu.

Flexible Job Shop Scheduling Problem (FJSSP) adalah sebuah pengembangan JSSP dimana setiap *job* memiliki urutan pemrosesan yang ditentukan melalui semua mesin yang tetap dan diketahui terlebih dahulu. FJSSP lebih komplek daripada JSSP karena selain mengurutkan sejumlah *n-job* yang akan diproses dalam sejumlah *m*-mesin, tetapi menentukan alokasi sebuah *job* yang akan diproses pada satu mesin dari beberapa alternatif mesin yang tersedia (Zhang, 2009). Masalah penjadwalan *job* dalam FJSSP dapat dipisahkan menjadi dua sub-masalah yaitu menentukan alokasi mesin untuk memproses *job* dari beberapa alternatif mesin yang tersedia dan mengurutkan operasi pada *job* yang telah dialokasikan ke salah satu mesin untuk mendapatkan jadwal *feasible* sesuai dengan tujuan yang akan dioptimalkan.

Pertama kali, Bruker dan Shile (1990) menyelesaikan FJSSP dengan dua *job.* Namun, permasalahan FJSSP secara umum merupakan kelompok permasalahan dimana belum ada algoritma yang dapat menemukan solusi optimal untuk permasalahan tersebut. Jika ada algoritma yang dapat menyelesaikan secara eksak namun akan menghabiskan waktu yang sangat lama apabila ukuran pencarian solusi permasalahan semakin besar. Untuk mengatasi keterbatasan metode eksak, dalam dua dekade terakhir ini metode metaheuristik menjadi sangat populer khususnya untuk menyelesaikan FJSSP karena dapat menyelesaikan permasalahan optimasi kombinatorial yang cukup rumit. Oleh karena itu, sebuah variasi metaheuristik dan pencarian lokal seperti *Simulated Annealing* (SA), *Tabu Search* (TS), algoritma Genetika, dan *Particle Swarm Optimization* (PSO) diterapkan untuk menyelesaikan permasalahan FJSSP dan menemukan penjadwalan/urutan yang optimal atau mendekati optimal (Bagheri, 2010).

Dalam FJSSP, metode metaheuristik dapat diklasifikasikan menjadi dua kategori utama yaitu pendekatan hirarki dan pendekatan terintegrasi. Dalam pendekatan hirarki, tugas mengalokasikan operasi untuk memproses *job* ke salah satu mesin yang tersedia dan mengurutkan operasi pada *job* yang telah dialokasikan ke salah satu mesin diperlakukan secara terpisah. Hal ini bertujuan untuk menguraikan masalah utama supaya waktu komplesitasnya semakin berkurang. Brandimarte (1993) adalah orang pertama yang menggunakan pendekatan ini untuk FJSSP. Dia menyelesaikan rute subpermasalahan menggunakan beberapa aturan penyelesaian yang ada dan kemudian fokus pada urutan subpermasalahan yang diselesaikan dengan algoritma *Tabu search* (TS). Xia dan Wu (2005) menggunakan pendekatan optimasi *hybrid Particle Swarm Optimization* (PSO) dengan *Simulated Annealing* (SA) untuk menyelesaikan *multi-objective* FJSP. Dalam pendekatannya, *Particle Swarm Optimization* (PSO) ditugaskan untuk mengalokasikan operasi ke sebuah mesin dari alternatif mesin yang tersedia dan algoritma *Simulated Annealing* (SA) untuk mengurutkan operasi pada *job.* Kacem (2002) menggunakan sebuah algoritma genetika dan *Approach Localization* dalam menyelesaikan FJSSP.

Pendekatan terintegrasi melakukan tugas mengalokasikan operasi ke salah satu mesin yang tersedia dan mengurutkan operasi pada *job* secara bersamaan dan biasanya memperoleh hasil yang lebih baik daripada pendekatan hirarki tetapi lebih banyak sulit untuk diselesaikan. Beberapa penelitian yang menggunakan pendekatan terintegrasi di antaranya adalah Chen dkk (1999) dan Jia (2003) dkk menggunakan algorima Genetika untuk menyelesaikan FJSP, *hybrid* PSO dengan *Tabu Search* dari Zhang dkk (2009) begitu juga untuk Larijani dkk (2010). Fattahi (2007) menggunakan sebuah model matematika dan dua pendekatan metaheuristik (*Simulated Annealing* dan *Tabu Search*).

Kasus FJSSP sudah menarik perhatian oleh penelitin baru-baru ini apalagi dalam sistem produksi dunia nyata yang menuntut fungsi tujuan yang ditentukan lebih dari satu tujuan atau *multi-objective*. Oleh karena itu, penelitian ini akan menggunakan sebuah pendekatan terintegrasi yang memanfaatkan algoritma imun untuk menyelesaikan *multi-objective* FJSP dengan fungsi tujuan meminimumkan makespan, waktu kerja terbesar mesin dan total waktu kerja terbesar mesin dari seluruh mesin. Sebuah metode pencarian lokal, *Simulated Anneling,* akan diterapkan untuk menghindari solusi optimal lokal dan mendapatkan hasil yang lebih baik.

Dalam FJSSP, terdapat sebuah himpunan *n job* $J=\{J_1,J_2,J_3,...,J_k,...,J_n\}$ yang akan di proses pada sebuah himpunan mesin $M=\{M_1,M_2,M_3,...,M_k,...,M_n\}$ dengan O_{ij} dan n_i masing-masing merupakan notasi operasi ke-*j* dari *job* ke-*i* dan jumlah operasi dari *job i*. Mesin yang mengoperasikan O_{ij} dinotasikan sebagai M_k dari himpunan mesin yang tersedia $M_{i,j}$ dimana $M_{i,j}$ merupakan notasi untuk sebuah himpunan mesin yang tersedia untuk O_{ij}. Waktu proses sebuah operasi O_{ij} pada mesin *k* dan dinotasikan dengan P_{ijk} . Terdapat dua jenis FJSP yaitu, FJSSP parsial apabilia terdapat $M_{i,j} \subset M$ untuk sedikitnya satu operasi dan FJSP total apabila terdapat $M_{i,j} = M$ untuk satu operasi (Kacem dkk, 2002) . Dalam penelitian ini, fungsi tujuan adalah meminimumkan kriteria sebagai berikut

1. Waktu penyelesaian mesin paling maksimal (*makespan*)
2. Waktu kerja maksimum dari sebuah mesin (*maximum machine workload*)
3. Total waktu kerja seluruh mesin (*Total workload of all machine*)

Menurut Larijani dkk (2010) dan Zhang G dkk, beberapa asumsi yang menyertai FJSSP adalah sebagai berikut:

1. Setiap operasi tidak dapat terganggu selama diproses pada mesin.
2. Setiap mesin dapat melakukan paling banyak satu operasi di suatu waktu pada satuan waktu
3. Kendala utama dari operasi dalam sebuah *job* dapat didefinisikan untuk setiap pasangan operasi.
4. Waktu penyiapan mesin dan pemindahan antar operasi dapat diabaikan.
5. Mesin independen dari satu sama lain, begitu juga untuk *job***.**
6. Tidak ada kendala utama antara operasi pada *job* yang berbeda
7. Tidak ada waktu luang dan tidak ada batas akhir keterlambatan yang ditetapkan
8. Semua *job* dan semua mesin tersedia pada saat *t*=0
9. Setiap mesin menjadi tersedia untuk operasi lain ketika operasi yang sedang diproses telah diselesaikan
10. *Job* dan mesin independen satu dengan lainnya.

Kemudian, contoh data dari FJSSP 4 *job* 5 mesin dengan 12 operasi dapat ditunjukkan dalam Tabel 1.

Tabel 1.
Data (FJSSP) 4 – *Job* 5 – Mesin dengan 12

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
Job_1	$O_{1,1}$	2	5	4	1	2
	$O_{1,2}$	5	4	5	7	5
	$O_{1,3}$	4	5	4	4	5

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5
Job_2	$O_{2,1}$	2	5	4	7	8
	$O_{2,2}$	5	6	9	8	5
	$O_{2,3}$	4	5	4	54	5
Job_3	$O_{3,1}$	9	8	6	7	9
	$O_{3,2}$	6	1	2	5	4
	$O_{3,3}$	2	5	4	2	4
	$O_{3,4}$	4	5	2	1	5
Job_4	$O_{4,1}$	1	5	2	4	12
	$O_{4,2}$	5	1	2	1	2

Berdasarkan pada Zhang dkk. (2009), model *multi-objective* FJSSP dengan kriteria meminimumkan waktu penyelesaian mesin paling maksimal, waktu kerja maksimum dari sebuah mesin dan total waktu kerja seluruh mesin secara berurutan adalah sebagai berikut

$$\min f_1 = \max_{1\le k\le m}\left(C_k\right) \quad (1)$$

$$\min f_2 = \max_{1\le k\le m}\left(W_k\right) \quad (2)$$

$$= \max_{1\le k\le m}\sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ijk}x_{ijk}$$

$$\min f_3 = \sum_{k=1}^{m} W_k \quad (4)$$

$$= \sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m} p_{ijk}x_{ijk}$$

Kendala

$$C_{ij} - C_{i(j-1)} \ge P_{ijk}x_{ijk}, j = 2,\ldots\ldots,n_i; \forall i,j \quad (5)$$

$$\left[\left(C_{hg} - C_{ij} - t_{hgk}\right)x_{hgk}x_{ijk} \ge 0\right] \vee \left[\left(C_{ij} - C_{hg} - t_{ijk}\right)x_{hgk}x_{ijk} \ge 0\right], \forall (i,j),(h,g),k$$

$$\sum_{k\in M_{ij}} x_{ijk} = 1, \forall\, i,j \quad (6)$$

dengan

$$x_{ijk} \begin{cases} 1, \text{ Jika mesin } k \text{ dipilih untuk operasi } O_{i,j} \\ 0, \text{ untuk yang lain.} \end{cases} \quad (7)$$

Keterangan

t_{ijk} = waktu awal dari operasi $O_{i,j}$ pada mesin k
$C_{i,j}$ = waktu selesai dari operasi $O_{i,j}$
i,h = indek *job* dengan $i,h = 1, 2, 3, \ldots n$
k = indek mesin, $k = 1, 2, 3, \ldots m$
j,g = indek urutan operasi dengan $j,g=1,2,..n_i$
C_k = waktu pnyelesaian dari mesin k
f_1 = waktu penyelesaian mesin paling maksimal
f_2 = waktu kerja maksimum dari sebuah mesin
f_3 = total waktu kerja seluruh mesin
W_k = workload atau total waktu M_k memproses operasi.

Pertidaksamaan (5) menyatakan bahwa operasi awal akan diproses terlebih dahulu. Pertidaksamaan (6) menyatakan bahwa setiap mesin dapat diproses hanya satu operasi untuk satu waktu. Pertidaksamaan (7) menyatakan status dimana satu mesin dapat dipilih dari kumpulan mesin yang tersedia dari setiap operasi.

METODE

Alogoritma Imun

Algoritma Imun merupakan metode yang dihasilkan melalui pengamatan terhadap teori immunologi, fungsi sistem imun manusia dan telah diterapkan untuk menyelesaikan beberapa permasalahan optimasi (Bondal, 2008; Chu dkk, 2008). Tujuan dari sistem imun adalah untuk melindungi tubuh manusia dari pathogen penyebab penyakit. Setiap sel dalam sebuah organisme terdiri dari gen dengan karakteristik tersendiri. Sistem imun yang komplek, robus, dan adaptif akan menggunakan mekanisme yang berbeda bahkan mengeliminasi sebuah sel dengan gen yang asing (patogen) tergantung pada seberapa banyak sel tersebut mempengaruhi sebuah organisme. Ketika sebuah patogen

menyerang sebuah organisme, sistem imun akan merangsang sel imun untuk merespon patogen sebagai antigen. Sekali antigen dikenali oleh sel imun, sebuah proses seleksi klon dari sel imun akan menentukan respon terhadap stimulus antigen dengan berkembang dan mensekresikan antibodi. Ketika terjadi proses reproduksi sel, sel telah mengamali berbagai gangguan dari patogen dan mengalami seleksi yang tinggi sehingga sel yang memiliki afinitas yang tinggi akan tetap bertahan dan mengeliminasi patogen dengan rasio mutasi yang rendah. Sementara itu, sel yang memiliki afinitas yang rendah akan mengalami rasio mutasi yang tinggi untuk memperoleh afinitas yang lebih tinggi.

Sebuah teknik kecerdasan buatan dalam bidang komputasi yang terinspirasi dari sistem imun dikenal sebagai algoritma imun (*artificial immune system*) dimana konsep dasar dari sistem imun diaplikasikan untuk menyelesaikan masalah optimasi, sains, teknik, dan lain sebagainya.

Prosedur yang digunakan dalam algoritma Imun berdasarkan Chu dkk, 2008 adalah sebagai berikut :

1. Mulai
   Mendefinisikan fungsi tujuan sebagai representasi dari antigen.
2. Inisialisasi
   Inisialisasi parameter jumlah iterasi yang dibutuhkan dan membangun *N* popsize secara random (solusi fisibel dari *multi-objective* FJSSP).
3. Evaluasi
   Nilai gaya kerekatan setiap antibody terhadap antigen (Afinitas Ag) akan dihutung berbasis pada nilai fungsi tujuan dari setiap antibodi.
4. Memori M
   Memilih p antibodi berdasarkan nilai fungsi tujuan terbaik sebagai himpunan memori M. Memori M analogi dari memori sel dalam sistem imun yang merupakan himpunan antibodi–antigen terbaik.
5. Klon
   Membentuk individu baru dengan mengkloning atau hasil salinan dari himpunan antibodi terbaik (memory M) secara proporsional sesuai dengan tingkat afinitas –Ag dari antibodi yang akan dikloning. Semakin tinggi tingkat afinitas –Ag maka semakin banyak antibodi baru yang dihasilkan dari proses kloning. Sebaliknya, semakin rendah tingkat gaya rekat –Ag maka semakin sedikit individu baru yang dihasilkan dari proses kloning.
6. Operasi Genetika
   Sebuah proses mengadopsi operasi dari algoritma generika yang didasarkan pada proses genetik pada mahluk hidup meliputi mutasi dan *crossover*
   a. Mutasi
      Mengubah sifat antibodi-antibodi hasil klon untuk menghasilkan antibodi dengan sifat yang berbeda.
   b. *Crossover*
      Pembentukan individu baru dengan persilangan genetik antar antibodi.
7. *Reselection*
   Memilih antibodi yang dihasilkan berdasarkan tingkat afinitas-Ag yang tinggi dan afinitas-Ab yang rendah sebanyak M ke himpunan memori M. Proses perhitungan afinitas –Ab dari setiap antibodi dapat menggunakan persamaan

$$\mathrm{Ab_j} = \frac{1}{1+d_j}, \quad j = 1, 2, ..., n \tag{8}$$

   dengan d_j merupakan Jarak Hamming antara antibodi j pada populasi dengan antibodi terbaik (*best antibody*) yang dirumuskan pada persamaan berikut

$$d_j = \sum_{i=1}^{k} \delta_i, \tag{9}$$

   dengan

$$\delta_i = \begin{cases} 1, & jika\ antibodi_j^i \neq antibodi_{best}^i \\ 0, & jika\ antibodi_j^i = antibodi_{best}^i \end{cases}$$

8. Kriteria Pemberhentian
   Proses akan terus berjalan sampai kriteria pemberhentian dipenuhi atau sebanyak iterasi yang dibutuhkan. Apabila kriteria pemberhentian belum terpenuhi maka kembali ke langkah 6.

Simulated Annealing

Algoritma *Simulated Annealing* merupakan algoritma yang terinspirasi dari proses *annealing* pada logam. *Annealing* adalah proses metalurgi dari sebuah material yang dipanaskan dengan suhu yang tinggi dan kemudian didinginkan perlahan-lahan. Ketika suhu menurun, pergerakan atom-atom itu semakin melemah dan bersamaan dengan itu atom-atom tersebut semakin tersusun secara teratur. Kemampuan eksplorasi simulated annealing pada proses penurunan suhu meningkatkan pencarian global dan kecepatan konvergensi untuk menghindari dari solusi optimal lokal (Wang dkk, 2008). Berikut ini adalah prosedur umum dari algoritma *Simulated Annealing*

1. Inisialisasi
   Menetapkan nilai suhu awal T_0, suhu akhir T_1, membangun sebuah solusi awal x_0 secara acak dan tetapkan x*= x_0 sebagai solusi kemudian hitung nilai fungsi tujuannya.
2. Membangun solusi baru
   Solusi baru dilakukan dengan membangun sebuah solusi ketetanggaan x' (*neighbourhood*) dari solusi awal x^* dan hitung nilai fungsi tujuannya .
3. Seleksi
   Proses seleksi dilakukan untuk menentukan sebuah solusi dari kandidat solusi yang ada (solusi awal dan solusi baru) dengan meminimalkan fungsi tujuan dan kondisi sebagai berikut
   - Jika $F(x') \leq F(x^*)$
     Tetapkan $x^*=x'$.
   - Jika $F(x') > F(x^*)$, tetapkan $x^*=x'$ dengan syarat sebuah bilangan random $r \in (0,1)$ kurang dari probabilitas $P=\min\{1,\exp((f(x^*)-f(x'))/T\}$
4. Update Suhu
   Turunkan T_0 suhu sebesar $T=T\times\lambda$ dengan $\lambda \in (0,1)$ koefisien penurunan suhu.
5. Kriteria Pemberhentian
   Jika suhu awal T_0 lebih besar dari suhu akhir T_1, maka proses akan dijalankan kembali pada langkah ke-2. Namun apabila terpenuhi kondisi sebaliknya atau suhu awal T_0 lebih kecil dari suhu akhir T_1 maka proses berhenti dengan mendapatkan penyelesaian dari hasil akhir solusi x^*.

Hybrid Algoritma Imun

Hybrid algoritma Imun dan *Simulated Anneling* merupakan algoritma penggabungan dari *Simulated Annealing* (SA) dan algoritma Imun. Proses *hybrid* dari kedua algoritma tersebut dilakukan dengan menyisipkan proses algoritma SA ke dalam algoritma Imun di mana SA melekat dalam algoritma Imun untuk meningkatkan kemampuan pencarian yaitu meningkatkan pencarian global dan kecepatan konvergensi untuk menghindari dari solusi optimal lokal (Wang dkk 2008). Proses SA dikerjakan setelah *crossover* dalam algoritma imun selesai dikerjakan. Hal ini dilakukan untuk mencari solusi baru yang diharapkan lebih baik dari individu baru hasil *crossover*. Penerimaan afinitas-Ag dari sebuah antibodi yang rendah dengan probabilitas tertentu dapat secara efisien melindungi para kandidat potensial yang mungkin dapat membawa ke solusi optimal global. *Hybrid* algoritma imun dengan *Simulated Annealing* dapat mendapatkan penyelesaian yang baik dalam memberikan solusi dan cocok untuk masalah multi-objective. Prosedur *hybrid* algoritma Imun dengan *Simulated Annealing* adalah sebagai berikut,

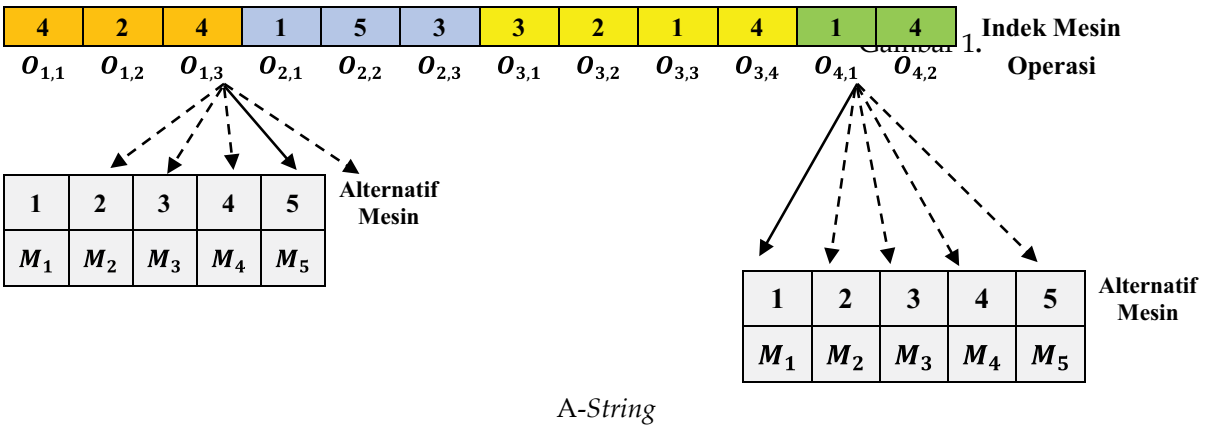
1. Inisialisasi
   Inisialisasi parameter yang dibutuhkan pada algoritma imun dan *Simulated Annealing* kemudian membangun *N* pasang antibodi sebanyak *popsize*
2. Evaluasi
   Mengevaluasi setiap antibodi berdasarkan fungsi tujuan (afinitas Ag)
3. Memori M
   Memilih antibodi terbaik sebanyak *p* pada memori M
4. Klon
   Membangun antibodi baru dengan menyalin antibodi dalam memori M secara proposional berdasarkan tingkat antibodi terbaik.
5. Operasi genetika

i. Mutasi
ii. *Crossover*

6. *Simulated Annealing*
   Setiap antibodi dalam memori M, hasil klon yang dimutasi dan *crossover* akan ditetapkan sebagai solusi awal dalam algoritma *Simulated Annealing*
   i. Membangun solusi baru dari masing-masing antibodi
   ii. Seleksi
   iii. Update Suhu
   iv. Kriteria Pemberhentian algoritma *Simulated Annealing*
7. *Reselection*
8. Kriteria Pemberhentian algoritma Imun

Pengkodean

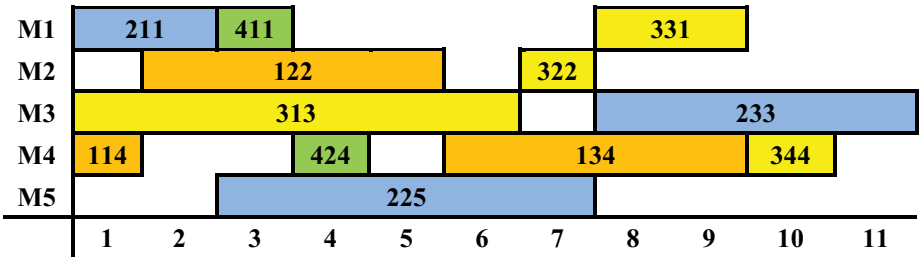
Zhang dkk. (2008) memisahkan FJSSP menjadi dua sub-masalah yaitu, mengalokasikan operasi dari *job* ke suatu mesin dari sekumpulan mesin yang tersedia dan mengurutkan operasi pada *job* yang telah dialokasikan ke salah satu mesin dengan urutan proses tertentu, dimana kedua sub-masalah tersebut secara berturut-turut direpresentasikan sebagai A-*String* dan B-*String* dengan jenis pengkodean permutasi (Gen dan Cheng, 1997). Keduanya merupakan satu pasang individu antibodi yang dibentuk dalam proses inisialisasi. Ilustrasi dari A-*String* dan B-*String* pada FJSSP 4 *job* dan 5 mesin dengan 12 operasi seperti contoh dataset pada Tabel 1 dapat ditunjukkan dalam Gambar 1 dan Gambar 2.



Gambar 1.
A-*String*

Gambar 2.
B-*String*

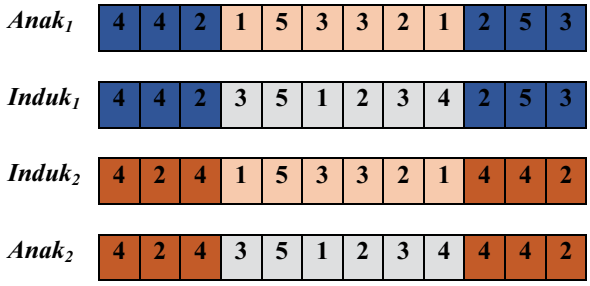
Berdasarkan pada Gambar di atas, $O_{i,j}$ menyatakan *job* ke-i dengan operasi ke-j. Indek dalam B-String dengan sebuah penendaan warna yang berbeda menunjukkan *job* ke-i terjadi n_i kali. Penjadwalan akan selalu fisibel jika masing-masing operasi dikawankan dengan indek *job* yang sesuai. Seperti diilustrasikan pada Gambar 2, sebuah urutan *job* dalam B-string, 1 – 2 – 3 – 1 – 4 – 2 – 4 – 1 – 3 – 3 – 2 – 3, akan dikaitkan dengan operasi yang bersesuaian sebagai $O_{1,1}$-$O_{2,1}$-$O_{3,1}$-$O_{1,2}$-$O_{4,1}$-$O_{2,2}$-$O_{4,2}$-$O_{1,3}$-$O_{3,2}$-$O_{3,3}$-$O_{2,3}$-$O_{3,4}$. Kemudian, setiap operasi tersebut dikaitkan ke setiap mesin yang bersesuaian pada A-*String* sehingga membentuk kode (1,1,4), (2,1,1),(3,1,3), (1,2,2), (4,1,1), (2,2,5), (4,2,4), (1,3,4), (3,2,2), (3,3,1), (2,3,3), (3,4,4). Sebagai contoh, alur kerja dari FJSSP 4 *job* 5 mesin dengan 12 operasi dapat digambarkan dengan menggunakan diagram *gantt* yang dapat ditunjukkan dalam Gambar 3.



Gambar 3.
Gantt Chart untuk FJSSP 4 – job 5 – mesin dengan 12 operasi

Operasi Crossover

Crossover adalah proses yang melibatkan dua buah antibodi untuk membentuk dua individu baru yang diharapkan dapat membentuk antibodi dengan hasil yang lebih baik. Dua tipe *crossover* akan diimplementasikan pada FJSSP diantaranya *Two Cut Point* Crossover (TCX) untuk A-*String* dan *Position Based Crossover* untuk B-*String*. Ilustrasi dari keduanya ditunjukkan dalam Gambar 4 dan Gambar 5 sebagai berikut



Gambar 4.
Two Cut Point Crossover (TCX)

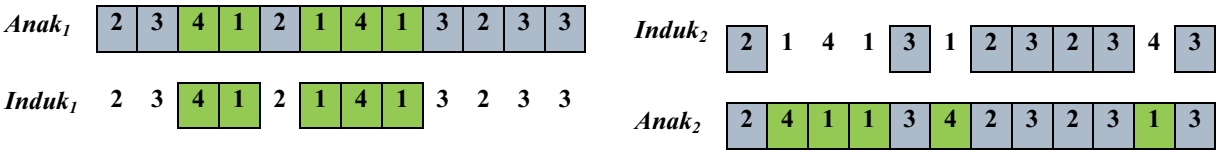
Dalam proses TCX, dua buah sub-string dari induk antibodi akan ditentukan secara acak kemudian mewariskannya secara bersilangan.

Sementara itu, Proses PBX membagi dua bagian misal J1 dan J2 dengan anggotanya ditentukan secara acak kemudian mewariskan sifat genetik tersebut (J1 untuk induk 1 dan J2 untuk induk 2) ke elemen antibodi baru (anak) yang bersesuaian. Kemudian, elemen yang masih kosong pada antibodi baru akan diisi dengan elemen induk secara bersilangan dari kiri ke kanan.

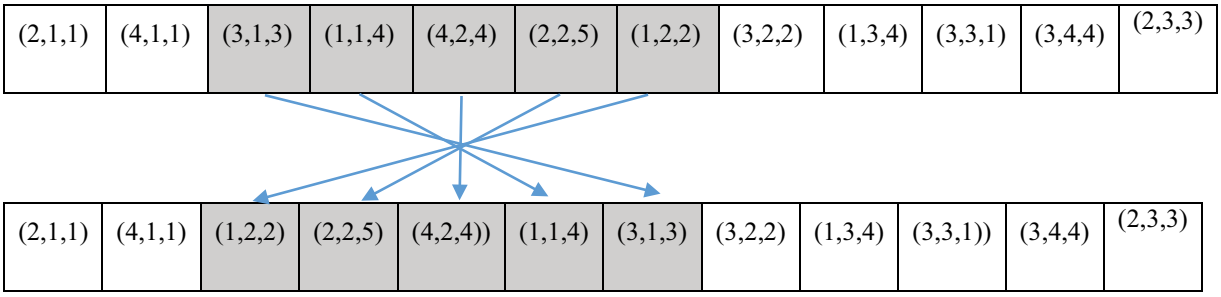
Operasi Mutasi

Mutasi dalam algoritma hanya akan diimplementasikan dalam B-*String* dengan jenis mutasi inversi (Gen dan Chen, 1997) yang diilustrasikan dalam Gambar 6. Dalam FJSSP, Mutasi digunakan untuk membentuk solusi ketetanggan untuk menghindari solusi yang terjebak dalam optimal lokal sehingga hanya akan mempengaruhi waktu penyelesaian mesin paling maksimal (makespan).

Berdasarkan pada Zhang dkk (2008), Jenis mutasi inversi dijalankan dengan membentuk sebuah sub-*string* kemudian membalik urutan dari kode sub-*string* tersebut.



Gambar 5.
Position Based Crossover



Gambar 6.
Mutasi inversi pada FJSSP

Afinitas-AG

Afinitas–Ag merupakan sebuah nilai hasil evaluasi dari setiap antibodi dengan fungsi tujuan gabungan dan yang telah diformulasikan pada persamaan (1), (2) dan (3). Berdasarkan Zhang (2008), Fungsi tujuan tersebut akan ditransformasikan dari permasalahan multi-objective ke mono-objective dengan penjumlahan bobot (weighted sum) dengan formulasi sebagai berikut:

$$\text{Min } F = w_1 x f_1 + w_2 x f_2 + w_3 x f_3 \qquad (10)$$

dengan $w_1, w_2, w_3 \in (0,1)$ dan $w_1+w_2+w_3=1$ adalah bobot masing-masing untuk *makespan*, *workload* terbesar dan total *workload* dari seluruh mesin yang dapat diberikan dengan nilai yang berbeda tergantung tujuan yang diprioritaskan.

HASIL DAN PEMBAHASAN

Untuk mengetahui kinerja *hybrid* algoritma Imun dalam menyelesaikan Multi-objective FJSSP, prosedur algoritma tersebut akan diimplementasikan dalam Java Netbean IDE dengan empat data uji coba yang diambil dari (Kacem dkk, 2002). Setiap data uji coba tersebut terdiri dari *job* yang memuat beberapa operasi dan mesin. Empat data uji coba tersebut meliputi 4 *job* 5 mesin dengan 12 operasi, 8 *job* 8 mesin dengan 27 operasi, 10 *job* 10 mesin dengan 30 operasi dan 15 *job* 10 operasi dengan 56 operasi.

Hasil *hybrid* algoritma dalam penelitian ini akan dibandingkan dengan penelitian sebelumnya yang ditunjukkan dalam Tabel 1, Tabel 2, Tabel 3 dan Tabel 4. Beberapa hasil penelitian sebelumnya meliputi *Temporal Decomposition* dari Kacem dkk (2002), algoritma Genetika Kacem dkk (2002), *Particle Swarm Optimization* (PSO) + *Tabu search* (TS) dari Zhang (2009) *Particle Swarm Optimization* (PSO)+*Simulated Annealing* (SA) dari Xia dan Wu (2005) *Approach Localization* (AL) dari Kacem dkk (2002) Algoritma Genetika atau *Genetic Algorithm* (GA)+ *Hill Climbing* (HC) dari Larijani dkk (2010) dan *Approach Localization* (AL) + *Co -evolutionary Genetic Algorithm* (CGA) dari Kacem dkk (2002). Dalam simulasi pada setiap data uji coba, masing masing parameter dalam *hybrid* algoritma Imun dengan nilai yang berbeda diproses 10 kali kemudian hasil terbaik diantaranya dipilih berdasarkan parameter yang proposional.

FJSSP 4 *Job* 5 Mesin dengan 12 Operasi

Sebuah evaluasi *hybrid* algoritma Imun sangat penting diketahui untuk mengetahui kefektifan algoritma dan kemampuan optimasinya. Proses simulasi dan uji coba FJSSP 4 *job* 5 mesin dengan 12 operasi dalam Tabel 1 dilakukan berdasarkan parameter *popsize* = 100; jumlah antibodi memori M = 50; suhu awal = 100 suhu akhir = 0 koefisien penurunan suhu = 0,1 probabilitas *crossover* () = 0,6; dan jumlah iterasi = 25. Hasil solusi terbaik dari *hybrid* algoritma Imun adalah

Makespan	11 satuan waktu
Maximum workload	10 satuan waktu
Total workload	32 satuan waktu

dengan *makespan* adalah waktu penyelesaian mesin paling maksimal, *maximum workload* adalah waktu kerja maksimum dari beberapa mesin dan Total *workload* adalah

total waktu kerja untuk seluruh mesin. Hasil penyelesaian tersebut dapat dibuat sebuah diagram *gantt* seperti pada Gambar 3.

Tabel 2.
Perbandingan hasil pada FJSSP 4 *job* 5 Mesin dengan 12 Operasi

Makespan		Max Workload	Total Workload
IA+SA	11	10	32
PSO+SA	11	10	32
GA+HC	12 13	8 7	32 33
AL+CGA	16	10	34

FJSSP 8 *Job* 8 Mesin dengan 27 Operasi

Tabel 4 menunjukkan FJSSP 4 *job* 5 mesin dengan 12 operasi. Proses simulasi dan uji coba berdasarkan parameter *popsize* = 300; jumlah antibodi memori M = 200; suhu awal = 100 suhu akhir = 0 koefisien penurunan suhu = 0,1 probabilitas *crossover* () = 0,5; dan jumlah iterasi = 200. Hasil solusi terbaik dari *hybrid* algoritma Imun adalah

Solusi 1

Makespan	15 satuan waktu
Maximum workload	12 satuan waktu
Total workload	75 satuan waktu

Solusi 2

Makespan	16 satuan waktu
Maximum workload	13 satuan waktu
Total workload	73 satuan waktu

dengan *makespan* adalah waktu penyelesaian mesin paling maksimal, *maximum machine workload* adalah waktu kerja maksimum dari sebuah mesin dan *total workload of all machine* adalah total waktu kerja seluruh mesin. Hasil penyelesaian tersebut dapat dibuat sebuah diagram *gantt* seperti pada Gambar 7

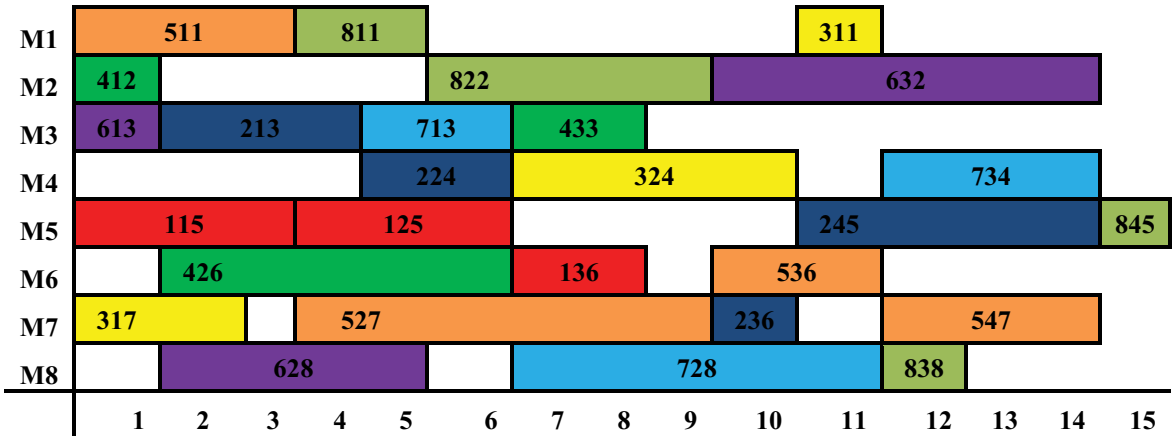
Tabel 3.
Perbandingan hasil pada FJSSP 8 *Job* 8 Mesin dengan 27 Operasi

Makespan		Max Workload	Total Workload
IA+SA	15 16	12 13	75 73
Approach Localization	16	13	75
GA+HC	14 14	11 12	78 75
AL+CGA	16	13	75
GA	16	11	77
Temporal Decomposition	19	19	91
PSO+SA	14 15	12 12	77 75

Tabel 4.
Data (FJSSP) 8 – *Job* 8 – Mesin dengan 12

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
Job_1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
Job_2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
Job_3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
Job_4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
Job_5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
Job_6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
Job_7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
Job_8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

Berdasarkan perbandingan hasil dengan algoritma lainnya, Tabel 3 menunjukkan bahwa *hybrid* algoritma Imun untuk FJSSP ukuran sedang, 8 *job* 8 mesin dengan 30 operasi, memperoleh hasil yang mendekati bahkan lebih baik dari algoritma lainnya. Begitu juga berlaku untuk perbandingan hasil FJSSP dengan algoritma lainnya ukuran kecil seperti 4 *job* 5 mesin dengan 12 operasi pada Tabel 2.



Gambar 7.
Diagram *gantt* hasil FJSSP 8 *job* 8 mesin dengan 27 operasi

FJSSP 10 *Job* 10 Mesin dengan 30 Operasi

Data uji coba skala menengah FJSSP 10 *job* 10 mesin dengan 30 operasi ditunjukkan pada Tabel 5. Proses simulasi dan uji coba berdasarkan parameter *popsize* = 600; jumlah antibodi memori M = 500; suhu awal = 100 suhu akhir = 0 koefisien penurunan suhu = 0,01 probabilitas *crossover* () = 0,6; dan jumlah iterasi = 500. Hasil solusi terbaik dari *hybrid* algoritma adalah

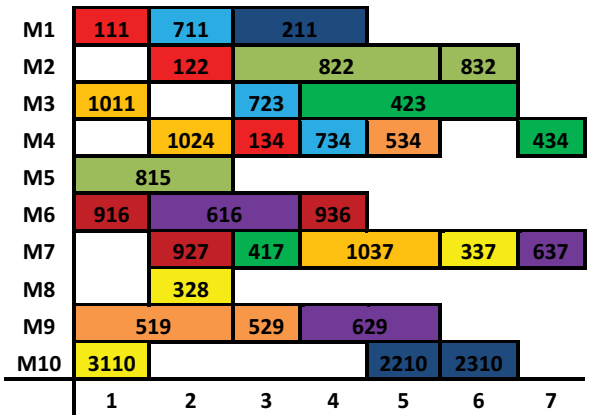
Makespan	7 satuan waktu
Maximum workload	6 satuan waktu
Total workload	42 satuan waktu

dengan *makespan* adalah waktu penyelesaian mesin paling maksimal, *maximum workload* adalah waktu kerja maksimum dari sebuah mesin dan total *workload* adalah total waktu kerja seluruh mesin. Hasil penyelesaian tersebut dapat dibuat sebuah diagram *gantt* seperti pada Gambar 8. Kemudian, Tabel 6 merupakan perbandingan hasil dengan algoritma lain yang diperoleh dari penelitian sebelumnya.

Tabel 5.
Data (FJSSP) 10 – *Job* 10 – Mesin dengan 30 Operasi dalam satuan waktu

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
Job_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3
Job_2	$O_{2,1}$	2	10	4	5	9	8	4	15	8	4
	$O_{2,2}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,3}$	6	11	2	7	5	3	5	14	9	2
Job_3	$O_{3,1}$	8	5	8	9	4	3	5	3	8	1
	$O_{3,2}$	9	3	6	1	2	6	4	1	7	2
	$O_{3,3}$	7	1	8	5	4	9	1	2	3	4

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
Job_4	$O_{4,1}$	5	10	6	4	9	5	1	7	1	6
	$O_{4,2}$	4	2	3	8	7	4	6	9	8	4
	$O_{4,3}$	7	3	12	1	6	5	8	3	5	2
Job_5	$O_{5,1}$	7	10	4	5	6	3	5	15	2	6
	$O_{5,2}$	5	6	3	9	8	2	8	6	1	7
	$O_{5,3}$	6	1	4	1	10	4	3	11	13	9
Job_6	$O_{6,1}$	8	9	10	8	4	2	7	8	3	10
	$O_{6,2}$	7	3	12	5	4	3	6	9	2	15
	$O_{6,3}$	4	7	3	6	3	4	1	5	1	11
Job_7	$O_{7,1}$	1	7	8	3	4	9	4	13	10	7
	$O_{7,2}$	3	8	1	2	3	6	11	2	13	3
	$O_{7,3}$	5	4	2	1	2	1	8	14	5	7
Job_8	$O_{8,1}$	5	7	11	3	2	9	8	5	12	8
	$O_{8,2}$	8	3	10	7	5	13	4	6	8	4
	$O_{8,3}$	6	2	13	5	4	3	5	7	9	5
Job_9	$O_{9,1}$	3	9	1	3	8	1	6	7	5	4
	$O_{9,2}$	4	6	2	5	7	3	1	9	6	7
	$O_{9,3}$	8	5	4	8	6	1	2	3	10	12
Job_{10}	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23



Gambar 8.
Diagram *gantt* hasil FJSSP 10 *job* 10 mesin dengan 30 operasi

Tabel 6.
Perbandingan hasil pada FJSSP 10 *Job* 10 Mesin dengan 30 Operasi

	Makespan	**Max Workload**	**Total Workload**
IA+SA	7	6	42
PSO+SA	7	6	44
GA+HC	7 8	5 5	43 42
AL+CGA	7	5	45
GA	7	7	53
PSO+TS	7 8	6 6	43 43
Approach Localizati-on	8	6	46
Temporal Decomposi-tion	16	16	59

Tabel 6 menunjukkan bahwa *hybrid* algoritma Imun pada FJSSP pada ukuran menengah dapat memperoleh hasil yang baik

FJSSP 15 *Job* 10 Mesin dengan 56 Operasi

Penelitian ini menggunaan data uji coba skala besar seperti FJSSP 4 *job* 5 mesin dengan 12 operasi seperti pada Tabel 7. Proses simulasi dan uji coba berdasarkan parameter *popsize* = 800; jumlah antibodi memori M = 600; suhu awal = 100 suhu akhir = 0 koefisien penurunan suhu = 0,01 probabilitas *crossover* () = 0,6; dan jumlah iterasi = 500. Hasil solusi terbaik dari *hybrid* algoritma adalah

Makespan	12 satuan waktu
Maximum workload	11 satuan waktu
Total workload	92 satuan waktu

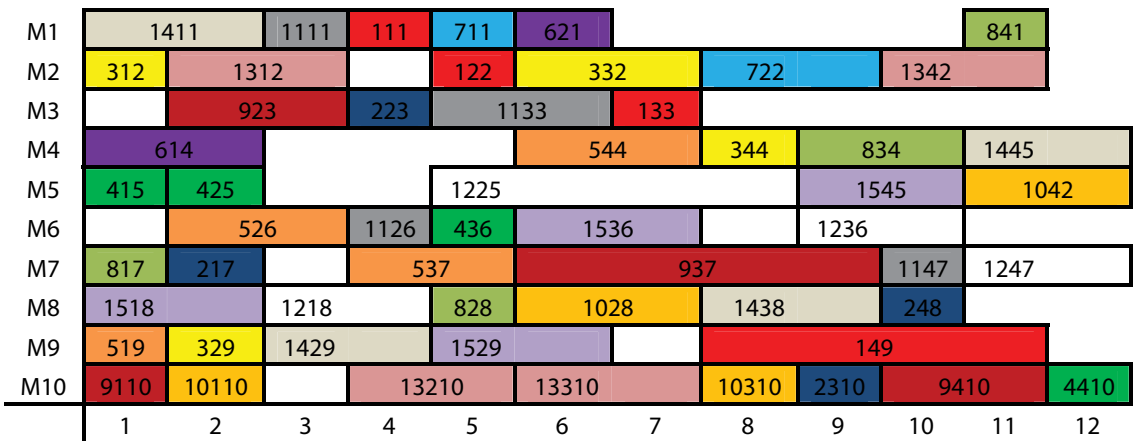
dengan *makespan* adalah waktu penyelesaian mesin paling maksimal, *maximum workload* adalah waktu kerja maksimum dari sebuah mesin dan total *workload* adalah total waktu kerja seluruh mesin. Hasil penyelesaian tersebut dapat dibuat sebuah diagram gantt seperti pada Gambar 9.

Tabel 7.
FJSSP 15 *Job* 10 Mesin dengan 56 Operasi dalam satuan waktu

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
Job_1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	4
	$O_{1,2}$	1	1	3	4	8	10	4	11	4	3
	$O_{1,3}$	2	5	1	5	6	9	5	10	3	2
	$O_{1,4}$	10	4	5	9	8	4	15	8	4	4
Job_2	$O_{2,1}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,2}$	6	11	2	7	5	3	5	14	9	2
	$O_{2,3}$	8	5	8	9	4	3	5	3	8	1
	$O_{2,4}$	9	3	6	1	2	6	4	1	7	2
Job_3	$O_{3,1}$	7	1	8	5	4	9	1	2	3	4
	$O_{3,2}$	5	10	6	4	9	5	1	7	1	6
	$O_{3,3}$	4	2	3	8	7	4	6	9	8	4
	$O_{3,4}$	7	3	12	1	6	5	8	3	5	2
Job_4	$O_{4,1}$	6	2	5	4	1	2	3	6	5	4
	$O_{4,2}$	8	5	7	4	1	2	36	5	8	5
	$O_{4,3}$	9	6	2	4	5	1	3	6	5	2
	$O_{4,4}$	11	4	5	6	2	7	5	4	2	1
Job_5	$O_{5,1}$	6	9	2	3	5	8	7	4	1	2
	$O_{5,2}$	5	4	6	3	5	2	28	7	4	5
	$O_{5,3}$	6	2	4	3	6	5	2	4	7	9
	$O_{5,4}$	6	5	4	2	3	2	5	4	7	5
Job_6	$O_{6,1}$	4	1	3	2	6	9	8	5	4	2
	$O_{6,2}$	1	3	6	4	4	7	5	4	6	5
Job_7	$O_{7,3}$	1	4	2	4	3	6	9	8	5	4
	$O_{7,4}$	2	2	4	4	2	3	5	4	2	5
Job_8	$O_{8,1}$	2	3	6	2	5	4	1	5	8	7
	$O_{8,2}$	4	5	6	2	3	5	4	1	2	5
	$O_{8,3}$	3	5	4	2	5	49	8	5	4	5
	$O_{8,4}$	1	2	36	5	2	3	6	4	11	2

Lanjutan Tabel 7

Job	$O_{i,j}$	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
Job_9	$O_{9,1}$	6	3	2	22	44	11	10	23	5	1
	$O_{9,2}$	2	3	2	12	15	12	12	14	18	16
	$O_{9,3}$	20	17	12	5	9	6	4	7	5	6
	$O_{9,4}$	9	8	7	4	5	8	7	4	56	2
Job_{10}	$O_{10,1}$	5	8	7	4	56	3	2	5	4	1
	$O_{10,2}$	2	5	6	9	8	5	4	2	5	4
	$O_{10,3}$	6	3	2	5	4	7	4	5	2	1
	$O_{10,4}$	3	2	5	6	5	8	7	4	5	2
Job_{11}	$O_{11,1}$	1	2	3	6	5	22	1	4	2	1
	$O_{11,2}$	2	3	6	3	2	1	4	10	12	1
	$O_{11,3}$	3	6	2	5	8	4	6	3	2	5
	$O_{11,4}$	4	1	45	6	2	4	1	25	2	4
Job_{12}	$O_{12,1}$	9	8	5	6	3	6	5	2	4	2
	$O_{12,2}$	5	8	9	5	4	75	63	6	5	21
	$O_{12,3}$	12	5	4	6	3	2	5	4	2	5
	$O_{12,4}$	8	7	9	5	6	3	2	5	8	4
Job_{13}	$O_{13,1}$	4	2	5	6	8	5	6	4	6	2
	$O_{13,2}$	3	5	4	7	5	8	6	6	3	2
	$O_{13,3}$	5	4	5	8	5	4	6	5	4	2
	$O_{13,4}$	3	2	5	6	5	4	8	5	6	4
Job_{14}	$O_{14,1}$	2	3	5	4	6	5	4	85	4	5
	$O_{14,2}$	6	2	4	5	8	6	5	4	2	6
	$O_{14,3}$	3	25	4	8	5	6	3	2	5	4
	$O_{14,4}$	8	5	6	4	2	3	6	8	5	4
Job_{15}	$O_{15,1}$	2	5	6	8	5	6	3	2	5	4
	$O_{15,2}$	5	6	2	5	4	2	5	3	2	5
	$O_{15,3}$	4	5	2	3	5	2	8	4	7	5
	$O_{15,4}$	6	2	11	14	2	3	6	5	4	8



Gambar 9.
Diagram gantt hasil FJSSP 15 *job* 10 mesin dengan 56 operasi

Tabel 8.
Perbandingan hasil pada FJSSP 15 *Job* 10 Mesin dengan 56 Operasi

	Makespan	Max Workload	Total Workload
IA+SA	12	11	92
PSO+ SA	12	11	91
GA+ HC	11 12	11 10	91 93
AL+ CGA	24 23	11 11	91 95
PSO+ TS	11 12	11 11	93 92

Berdasarkan perbandingan hasil dengan algoritma lainnya, Tabel 8 menunjukkan sebuah keefektifan *hybrid* algoritma Imun pada FJSSP ukuran besar, 15 *Job* 10 Mesin dengan 56 Operasi, yang ditunjukkan dengan hasil yang sebanding dengan hasil yang diperoleh dari algoritma lainnya.

SIMPULAN

Dalam penelitian ini, sebuah *hybrid* algoritma Imun dengan *Simulated Annealing* digunakan untuk menyelesaikan *multi-objective flexible job shop scheduling problem* dengan fungsi tujuan meminimalkan waktu penyelesaian mesin paling maksimal (*makespan*), waktu kerja maksimum dari sebuah mesin (*maximum machine workload*) dan total waktu kerja seluruh mesin (*Total workload of all machine*). Fungsi tujuan tersebut ditransformasikan dari permasalahan *multi-objective* ke *mono-objective* dengan penjumlahan bobot (*weighted sum*). Karena kompleksitas permasalahan tersebut, algoritma Imun yang dikombinasikan *Simulated Annealing* digunakan sebagai pencarian lokal sehingga memperoleh penyelesaian yang lebih baik. Hasil penyelesaian menggunakan *hybrid* algoritma Imun dalam penelitian ini dibandingkan dengan hasil penyelesaian *multi-objective* FJSSP dari penelitian sebelumnya dan menunjukkan hasil yang efektif dan efisien dalam menyelesaikan *multi-objective* FJSSP.

DAFTAR PUSTAKA

Bagheri, A. Zandieh, M. Mahdavi, I. Yazdani, M. 2010. An Artificial Immune Algorithm for The Flexible Job Shop Scheduling Problem. *Future Generation Computer Systems* 26. 533–541.

Baykasoglu, A. Ozbakir, L. dan Sonmez, A.I. 2004. Using multiple objective tabu search and grammars to model and solve multi-objective flexible job-shopscheduling problems. *Journal of Intelligent Manufacturing*. 15(6): 777–785.

Bondal, A. 2008. Artificial Immune System Applied to Job Shop Scheduling. *Master Thesis*. Russ College of Engineering and Technology of Ohio University.

Brandimarte, P. 1993. Routing and Scheduling in a Flexible Job Shop by Taboo Search. *Annals of Operation Research* 41. 157–183.

Bruker, P. dan Schlie, R. 1990. Job Shop Scheduling with multi-purpose machine. *Computing* 45. 369–375

Chaudhry, I.A. Khan, A.M. dan Khan, A.A. 2013. A Genetic Algorithm for Felxible Job Shop Scheduling. *Proceeding of the World Congress on Engineering* I. 1–6.

Chen, L. Ihlow, J. Lehmann, C. 1999. A Genetic Algorithm for Flexible Job Shop Scheduling. *IEEE Internasional Conference on Robotics and Automation*. Detroit. 1120–1125.

Chibante, R. 2010. *Simulated Annealing Theory with Application*. Sciyo. Rijeka. Croatia.

Chu, C.W. Lin, M.D. Liu, G.F. dan Sung, Y.H. 2008, *Application of Immune Algorithms on Solving Minimum-Cost Problem of Water Distribution Network*. *Mathematical and Computer Modelling* 48(11-12). 1888–1900.

Fattahi, P. Saidi, M.M. Jolai, F. 2007. Mathematical Modeling and Heuristic Approach to Flexible Job Shop Scheduling Problem. *Kournal of*

Intelligent Manufacturing 18(3). 331–342.

Gen, M. dan Cheng, R. 1997. Genetic Algorithms and Engineering Design. John Wiley and Sons. New York.

Jia, H.Z. Nee, A.Y.C. Fuh, J.Y.H. Zhang, Y.F. 2003. A Modified Genetic Algorithm for Distributed Scheduling Problems. *Internasional Journal of Intelligent Manufacturing* 14. 351–362.

Kacem, I, Hammadi, S. Borne, P. 2002. Approach by Localization and Multi-Objetive Evolutionary Optimization for Flexible Job Shop Scheduling Problem. IEEE Transaction on Systems, Man, and Cybernetics, Part C 32 (1). 1–13.

Larijani, A.M. Laghaie, K.S. Heydari, M. 2010. Solving Flexible Job Shop Scheduling with Multi Objective Approach. *Internasional Journal of Industrial Engineering & Production Research* 21(4). 197–209.

Wang, X. Gao, X.Z. Ovasca S.J. 2008. A Simulated Annealing-Based Immune Optimization Method. *Proceedings of the 2nd International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning*. 41–47.

Xia, W. J. dan Wu, Z.M. 2005. An effective hybrid optimization approach for multiobjective flexible job-shop scheduling problems. *Computers and Industrial Engineering* 48(2). 409–425.

Zhang, G. Shao, X. Li, P. Gao, L. 2009. An Effective Hybrid Particle Swarm Optimization Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem. Computers & Industrial Engineering. 56(4). 1309-1318