Implementasi *Differential Evolution* untuk Optimasi Jadwal Produksi

Hendry Setiawan[1], Dewi Fandelia Tan [2], Kestrilia Rega Prilianti[3]
[1,2,3] Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Ma Chung
Jl. Villa Puncak Tidar N-01, Malang 65151, Jawa Timur, Indonesia
Email: [1]hendry.setiawan@machung.ac.id, [2]dewitan01@gmail.com, [3]kestrilia.rega@machung.ac.id

Abstract. *Scheduling is one of the important things that needs to be considered by any company. A good scheduling process can improve the effectiveness and efficiency of production systems at the company. The problem often occured in this field is a delay in working orders from the deadline and the production of defective products (afal) is still many. The limited ability of the operator to divide the job on the machine causes mistakes of scheduling. To solve that problem, an application was developed to optimize the production schedule automatically. This application was made by implementing the differential evolution algorithm. This production scheduling solution will be represented in vector form. Each vector will be calculated fitness value by the criteria of time and minimum afal. This process will achieve the best vector that provides an optimal schedule. The testing results show that the application can optimize the schedule of production with the average of accuracy rate reaching 99,54%. The scheduling results using differential evolution algorithm can reduce 8,19% of afal, and 97,51% of computing time.*
Keywords: *afal, differential evolution, schedulling*

Abstrak. *Penjadwalan merupakan salah satu hal penting yang perlu diperhatikan oleh setiap perusahaan. Penjadwalan yang baik akan meningkatkan efisiensi dan efektifitas dari perusahaan tersebut. Permasalahan yang sering terjadi adalah masih sering terjadinya keterlambatan dalam pengerjaan pesanan dari batas waktu yang ditentukan dan kerusakan produksi (afal) yang dihasilkan masih sangat tinggi. Hal ini dikarenakan terbatasnya kemampuan operator dalam membagi job pada mesin, sehingga memungkinkan terjadinya kesalahan penjadwalan. Untuk memecahkan permasalahan tersebut, dibuat sebuah aplikasi yang dapat melakukan optimasi jadwal produksi secara otomatis. Aplikasi ini dibuat dengan mengimplementasikan algoritma differential evolution. Solusi penjadwalan produksi ini akan direpresentasikan dalam bentuk vektor. Setiap vektor akan dihitung nilai fitness dengan kriteria minimasi waktu dan afal. Proses ini akan dilakukan hingga mencapai vektor terbaik yang mampu memberikan jadwal yang optimal. Hasil pengujian yang dilakukan menunjukkan bahwa aplikasi dapat melakukan optimasi jadwal produksi dengan rata-rata tingkat keakuratan mencapai 99,54%. Hasil penjadwalan menggunakan algoritma differential evolution dapat menurunkan afal sebesar 8,19%, dan waktu komputasi sebesar 97,51%.*
Kata Kunci: *afal, algoritma differential evolution, penjadwalan*

1. Pendahuluan

Kantung plastik diproduksi berdasarkan pada kriteria yang diinginkan seperti jenis, lebar, serta ketebalan plastik. Dalam survei didapati 3 jenis plastik yang diproduksi yaitu *polypropylene* (PP), *polyethylene* (PE), dan *High Density Polyethylene* (HDPE). Dari ketiga jenis plastik tersebut yang paling jenis PP adalah jenis yang paling diminati. Pengelolaan plastik jenis PP ini menggunakan 17 mesin yang terdiri enam mesin plastik kecil, sembilan mesin plastik sedang, dan dua mesin plastik besar yang dibedakan berdasarkan ukuran serta kapasitas produksi dalam satu hari. Mesin kecil berkapasitas memproduksi maksimal 400 kg plastik tiap hari. Mesin sedang memiliki kapasitas maksimal 700 kg plastik dan mesin besar berkapasitas 1200 kg tiap harinya. Proses awal dalam menghasilkan kantung plastik yang sesuai dengan ukuran dan keinginan selalu diawali dengan adanya kantung plastik yang belum stabil atau dikenal dengan kerusakan produksi (afal). Order yang diterima (*job*) perusahaan dalam satu hari

beragam, khusus untuk plastik jenis PP, keseluruhan jumlah *job* dalam satu minggu mencapai hingga diatas 4000kg. Dengan pertimbangan reduksi afal dan pengerjaan oleh satu mesin, pekerjaan akan selesai dengan waktu yang lebih lama. Apabila proses kerja diselesaikan dengan penyebaran pekerjaan kepada beberapa mesin, maka waktu produksi akan menjadi lebih singkat namun peningkatan afal terjadi. Produksi kantung plastik yang dilakukan tanpa memperhatikan batas waktu dan afal berdampak negatif pada perusahaan, oleh karena itu dibutuhkan penjadwalan.

Penjadwalan yang baik merupakan salah satu hal penting yang perlu diperhatikan oleh setiap perusahaan untuk meningkatkan efektivitas dan efisiensi sistem produksi pada perusahaan tersebut. Penjadwalan dapat diartikan sebagai proses pengaturan alokasi sumber daya untuk menyelesaikan tugas-tugas yang melibatkan pekerjaan, sumber daya, dan waktu. Penjadwalan melibatkan pengaturan sumber daya dalam menyelesaikan tugas yang meliputi waktu, pekerjaan dan sumber daya [1]. Permasalahan penjadwalan melibatkan keputusan untuk menentukan urutan pekerjaan yang akan dikerjakan dahulu dan sekaligus berpengaruh terhadap waktu penyelesaiannya (*makespan*) [2]. Dengan adanya proses penjadwalan yang baik, maka kinerja mesin-mesin dapat dimanfaatkan dengan baik pula sehingga mengurangi mesin-mesin yang menganggur. Penjadwalan pada proses produksi *job shop* merupakan proses pengurutan pekerjaan yang menyusun semua operasi dari semua *job* pada tiap mesin sehingga semua *job* dapat diproses.

Permasalahan yang terjadi di lapangan ialah sering terjadinya keterlambatan dalam pengerjaan *job* dari batas waktu yang ditentukan dan meminimalkan *afal* yang dihasilkan. Hal ini terjadi karena kurangnya kemampuan operator dalam melakukan penjadwalan secara manual yang dilakukan berdasarkan pada urutan *job* yang diterima. Mekanisme selanjutnya yang dirancang adalah membuat penjadwalan berdasarkan penerimaan *job* secara keseluruhan yang diterima dalam satu hari dan diselesaikan dengan bantuan algoritma *searching.*

Beberapa penelitian telah berhasil membuktikan bahwa permasalahan penjadwalan *job shop* mampu dioptimalkan menggunakan pendekatan *metaheuristik* seperti Algoritma Genetika [3], Penjadwalan yang melibatkan waktu, pekerjaan dan sumber daya dengan Algoritma *Artificial Immune System* [1], penjadwalan *job shop* dengan Algoritma *Simulated Anneling* [4]. Dalam penjadwalan yang melibatkan banyak mesin [5], salah satu algoritma pencarian yang memiliki kemampuan sebagai metode optimasi global optimum yang efektif ialah algoritma *Differential Evolution* (DE). Algoritma DE merupakan algoritma yang pencarian yang cepat dan handal dalam menyelesaikan permasalahan numerik serta menemukan penyelesaian global optimum [6]. Penyelesaian permasalahan *Flexible Job Shop Schedulling Prolem* dengan fungsi obyektif meminimumkan *makespan* dengan menggunakan algoritma *Differential Evolution* [7].

Dari Latar belakang yang telah dijelaskan sebelumnya muncul perumusan masalah yaitu bagaimana merancang sistem otomasi pada penjadwalan produksi dengan memanfaatkan algoritma pencariaan *Differential Evolution*.

2. Tinjauan Pustaka

Algoritma *Differential Evolution* (DE) merupakan salah satu dari algoritma metaheuristik yang digagas oleh Storn dan Price pada tahun 1997. Algoritma DE termasuk dalam keluarga *Evolutionary Algorithm* (EA) yaitu *evolutionary population-based algorithm* dimana prinsip dan filosofi algoritmanya meniru perilaku evolusi biologi. Dalam prakteknya teknik optimasi terdiri dari tiga hal [8]. Pertama, pencarian harus mendapatkan nilai global optimum. Kedua, konvergensinya cepat. Ketiga, aplikasi menyediakan beberapa kontrol untuk parameter yang bertujuan memudahkan penggunaan. Kemunculan DE didasarkan pada ketiga persyaratan tersebut sehingga DE memiliki keunggulan yaitu konsep yang sederhana, implementasi yang mudah dan konvergensi yang cepat, walaupun kinerja DE sangat tergantung dari parameternya [9]. Langkah-langkah algoritma DE meliputi inisialisasi populasi, mutasi, *crossover* dan *selection*.

Inisialisasi Populasi*:* Sebelum melakukan inisialisasi titik populasi, perlu dilakukan penentuan batas atas (ub) dan batas bawah (lb). Berikutnya adalah membangkitkan bilangan

acak untuk setiap parameter j dari vektor i pada iterasi g. Misal nilai inisial (g = 0) dapat diwakili dengan notasi sebagai berikut:

$$x_{j,i,0} = lb_j + rand_j(0,1)(ub_j - lb_{j)}) \quad (1)$$

Bilangan *random* dibangkitkan dengan fungsi *rand* (), dimana bilangan yang dihasilkan terletak pada rentang (0,1).

Mutasi*:* mutasi dilakukan dengan mengambil 3 buah vektor dari populasi (r1, r2, dan r0). Perbedaan 2 vektor pertama yaitu r1 dan r2 akan menghasilkan vektor d. Vektor d selanjutnya akan dikalikan dengan konstanta mutasi dan ditambahkan pada vector r0 [10]. Berikut ini adalah persamaan yang menunjukkan bagaimana membentuk vektor mutan *($v_{i,g}$)*:

$$v_{i,g} = x_{r0,g} + F\left(x_{r1,g} - x_{r2,g}\right) \quad (2)$$

Dimana *r*0, *r*1, *r*2 merupakan indeks acak, bilangan integer yang berbeda.

Crossover: Pada tahap ini DE menyilangkan setiap vektor, $x_{i,g}$ dengan vektor mutan, $v_{i,g}$ untuk membentuk vektor hasil persilangan yaitu $u_{i,g}$. Persamaan untuk vektor uji adalah sebagai berikut:

$$u_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & if\ (rand)\ j(0,1) \le Cr, or \quad j = rand \\ x_{j,i,g} & sebaliknya \end{cases} \quad (3)$$

Probabilitas crossover, *Cr* ϵ (0,1) merupakan nilai yang didefinisikan untuk mengendalikan fraksi nilai parameter yang disalin dari mutan.

Selection: Apabila vektor trial $U_{i,g}$, memiliki fungsi evaluasi yang kurang dari fungsi evaluasi vector target $x_{i,g}$ maka pada generasi berikutnya yang terpilih adalah $u_{i,g}$ dan sebaliknya [5]. Proses ini dilakukan untuk seluruh vektor dalam satu populasi.

$$x_{i,g+1} = \begin{cases} u_{i,g} & if\ \ f\left(u_{i,g}\right) \le f\left(x_{i,g}\right) \\ x_{i,g} & sebaliknya \end{cases} \quad (4)$$

Proses akan diulang sampai *stopping criterion* tertentu dicapai.

3. Metodologi Penelitian

3.1. Analisa

Dalam proses produksi *job shop*, satu pekerjaan dapat dikerjakan pada satu atau beberapa mesin yang memiliki pemrosesan yang sama ataupun berbeda. Kesalahan penentuan prioritas *job* yang akan dikerjakan pada setiap mesin akan mempengaruhi waktu penyelesaian pekerjaan-pekerjaan yang lainnya serta afal yang dihasilkan. Hal ini akan menimbulkan keterlambatan penyelesaian produksi serta tingginya afal yang dihasilkan.

Selama ini proses penjadwalan yang dilakukan masih secara manual. Hal ini dikarenakan belum adanya fasilitas yang mampu secara otomatis melakukan optimasi pada penjadwalan yang dapat meminimalkan keterlambatan serta afal yang dihasilkan. Penjadwalan dikerjakan hanya berdasarkan perkiraan operator bagian penjadwalan sesuai urutan kedatangan pesanan.

Berdasarkan hasil pengamatan, diketahui bahwa selalu terdapat afal dalam setiap proses mengerjakan order. Untuk sekali pengerjaan order, afal dapat mencapai sekitar ± 5 kg untuk mesin kecil, 7 kg untuk mesin sedang, dan 10 kg untuk mesin besar. Afal selalu terjadi diawal proses pengerjaan order.

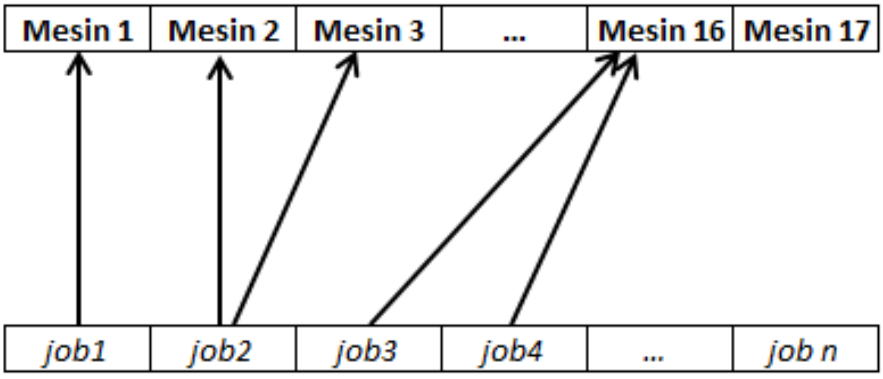
3.2. Kebutuhan Data

Data masukan pada perancangan aplikasi ini berupa pesanan kantung plastik/*job* dari pelanggan. Keseluruhan data *job* yang dikumpulkan berasal dari data *job* selama tujuh hari yang akan digunakan untuk uji coba keberhasilan algoritma *differential evolution*. Data pesanan tersebut meliputi jenis produk apa saja yang dipesan, ukuran pesanan, jumlah pesanan (dalam satuan kilogram), serta batas waktu yang diberikan oleh konsumen untuk menyelesaikan pesanan (*due date*). Berikut contoh *job* dari pelanggan dalam satu minggu yang melibatkan mesin kecil, mesin sedang, dan mesin besar, dapat dilihat pada Tabel 1. Data pesanan pada Tabel 1 akan digunakan untuk uji coba parameter terbaik dan uji coba performa penjadwalan yang dihasilkan oleh aplikasi menggunakan algoritma *differential evolution*.

Tabel 1. Sejumlah Job dalam Satu Minggu

No.	Jenis Produk	Ukuran (µ x cm)	Jumlah (kilogram)
1	Polos Rol PP	50x10	100
2	Polos Rol PP	50x10	200
3	Polos Pot PP	25x25	500
4	Polos Pot PP	25x33	400
5	Polos Pot PP	40x42	200
6	Polos Pot PP	40x42	300
7	Polos Pot PP	10x20	100
8	Polos Pot PP	60x50	200
9	Polos Pot PP	60x50	100
10	Polos Pot PP	60x58	1000
11	Polos Pot PP	10x20	300
12	Polos Pot PP	50x10	300
13	Polos Pot PP	10x18	500

Model pembagian *job* yang dilakukan secara manual seperti pada Gambar 1.



Gambar 1. Pembagian *job* pada mesin

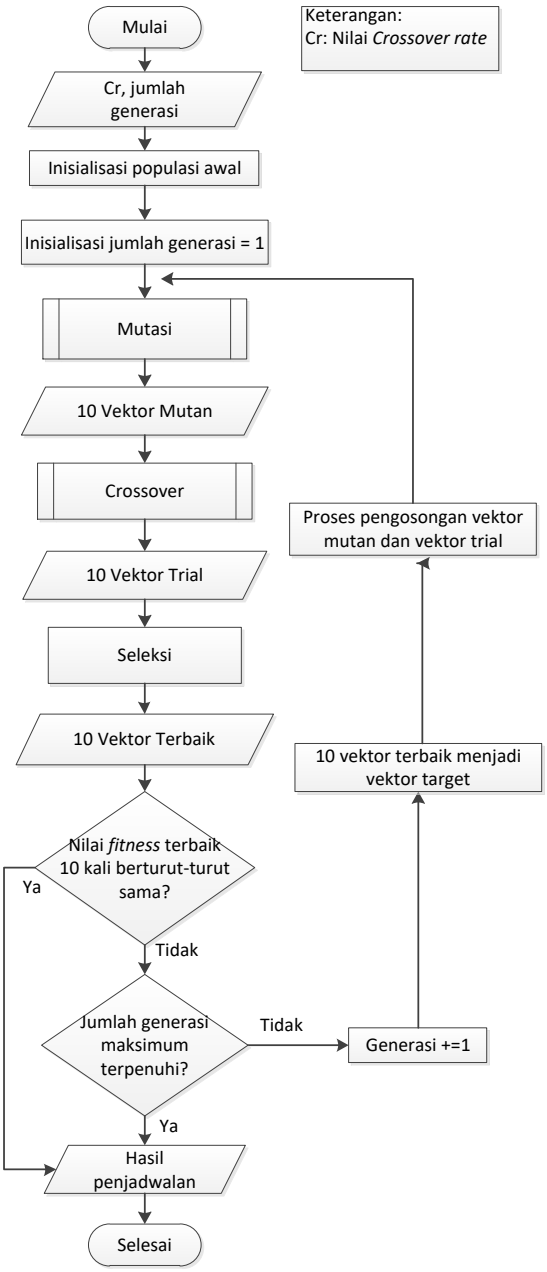
Pengerjaan *job* untuk kantong plastik ini dikerjakan pada divisi *extruder* saja dengan melibatkan mesin yang dimiliki sebanyak 17 mesin untuk memproduksi kantong plastik jenis PP. Spesifikasi untuk masing-masing mesin dilihat pada Tabel 2.

Tabel 2. Spesifikasi mesin produksi kantung plastik jenis pp

No. Mesin			Ukuran		Kapasitas (kg)/hari
			minimal	maksimal	
1	PP	S	20 x 10	40 x 18	700
2	PP	S	20 x 10	40 x 18	700
3	PP	S	20 x 10	40 x 18	700
4	PP	K	20 x 8	40 x 22	400
5	PP	K	30 x 7	40 x 22	400
6	PP	S	20,25 x 15	50 x 35	700
7	PP	K	20 x 6	40 x 23	400
8	PP	K	20 x 6	40 x 23	400
9	PP	K	20 x 6	40 x 23	400
10	PP	S	20,25 x 15	50 x 39	700
11	PP	K	20 x 6	40 x 23	400
12	PP	B	30 x 30	60 x 55	1200
13	PP	S	20 x 15	40 x 40	700
14	PP	S	20 x 15	60 x 36	700
15	PP	S	20 x 15	40 x 40	700
16	PP	B	20 x 30	80 x 65	1200
17	PP	S	20 x 10	40 x 18	700

3.3 Rancangan Aplikasi

Alur sistem penjadwalan tersebut terdiri dari beberapa langkah utama. Pertama, ketika aplikasi dijalankan, daftar order pada hari tersebut akan ditampilkan. Kedua, daftar order tersebut akan diolah dengan dilakukan penambahan, atau pengubahan data. Ketiga, daftar order tersebut mendapatkan optimasi jadwal produksi dengan menggunakan Algoritma *Differential Evolution*. Keempat, setelah proses optimasi selesai, hasil penjadwalan akan ditampilkan. Apabila *user* telah cukup puas dengan jadwal yang dihasilkan, jadwal akan dikirim ke dalam file excel. Setelah proses pengolahan data selesai, daftar pesanan pada hari tersebut dapat dijadwalkan. Sebelum dilakukan optimasi penjadwalan, data pesanan yang telah diolah sebelumnya akan ditampilkan pada *listview* yang mengelompokkan pesanan-pesanan sesuai kategori dan dimulai dari batas waktu penyerahan yang paling dekat. Selain itu, *listview* tersebut juga menampilkan nomor Perintah Kerja (PK) bagi setiap pesanan yang dikelompokkan berdasarkan lebar dan kategori yang sama. Alur dan proses *differential evolution* dapat dilihat pada Gambar 2.



Gambar 2. Diagram Alir Differential Evolution

Proses ini akan dimulai dari inisialisasi populasi awal sejumlah 10 vektor sebagai representasi solusi dari permasalahan atau disebut juga sebagai vektor target. Agar mendapatkan jadwal yang diharapkan, maka dimodelkan dalam bentuk *slot-slot* yang setiap *slot*-nya merepresentasikan beban sebesar 100 kg. Setiap kapasitas mesin akan dimodelkan dalam bentuk *slot-slot* dimana setiap *slot* akan diberikan nomor urut. Nomor urut ini berfungsi untuk mengetahui batasan ukuran tiap mesin. Berikut adalah tabel pemodelan dalam bentuk *slot* untuk mesin kecil, sedang, dan besar yang dapat dilihat pada Tabel 3, Tabel 4, dan Tabel 5.

Tabel 3. Model slot untuk mesin kecil

No Mesin	Hari ke -1				Hari ke-2				…	Hari ke-7			
	100	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**…**	**100**	**100**	**100**	**100**
1	0	1	2	3	23	25	26	27	…	144	145	146	147
2	4	5	6	7	28	29	30	31	…	148	149	150	151
3	8	9	10	11	32	33	34	35	…	152	153	154	155

4	12	13	14	15	36	37	38	39	…	156	157	158	159
5	16	17	18	19	40	41	42	43	…	160	161	162	163
6	20	21	22	23	44	45	46	47	…	164	165	166	167

Tabel 4. Model slot untuk mesin sedang

No Mesin	**Hari ke -1**								**Hari ke-7**						
	100	**100**	**100**	**100**	**100**	**100**	**100**	**…**	**100**	**100**	**100**	**100**	**100**	**100**	**100**
7	168	169	170	171	172	173	174	…	546	547	548	549	550	551	552
8	175	176	177	178	179	180	181	…	553	554	555	556	557	558	559
9	182	183	184	185	186	187	188	…	560	561	562	563	564	565	566
10	189	190	191	192	193	194	195	…	567	568	569	570	571	572	573
11	196	197	198	199	200	201	202	…	574	575	576	577	578	579	580
12	203	204	205	206	207	208	209	…	581	582	583	584	585	586	587
13	210	211	212	213	214	215	216	…	588	589	590	591	592	593	594
14	217	218	219	220	221	222	223	…	595	596	597	598	599	600	601
15	224	225	226	227	228	229	230	…	602	603	604	605	606	607	608

Tabel 5. Model slot untuk mesin besar

No Mesin	**Hari ke -1**													**Hari ke-7**
	100	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**…**	**100**
16	609	610	611	612	613	614	615	616	617	618	619	620	…	…
17	621	622	623	624	625	626	627	628	629	630	631	632	…	776

Berdasarkan pemodelan *slot-slot* tersebut, bentuk vektor solusi dapat direpresentasikan Nilai pada vektor didapatkan secara *random* tetapi tidak menyimpang dari batasan *slot-slot* yang disesuaikan dengan *upper bound* (UB) dan *lower bound* (LB) dari masing-masing jenis mesin. Contoh representasi vektor sesuai jenis mesin dapat dilihat pada Tabel 6, Tabel 7, dan Tabel 8.

Tabel 6. Representasi vektor mesin kecil

Jenis Mesin	**Mesin Kecil**						
Id Order			0101		0101	0101	0101
Id Slot	0(LB)	…	…	…	…	…	167(UB)
Hari ke-	1	2	3	4	5	6	7

Tabel 7. Representasi vektor mesin sedang

Jenis Mesin	**Mesin Sedang**						
Id Order	0201		0201				0201
Id Slot	168(LB)	…	…	…	…	…	608(UB)
Hari ke-	1	2	3	4	5	6	7

Tabel 8. Representasi vektor mesin besar

Jenis Mesin	**Mesin Sedang**						
Id Order		301			301		
Id Slot	609(LB)	…	…	…	…	…	776(UB)
Hari ke-	1	2	3	4	5	6	7

Setelah inisialisasi vektor target, tahap dilanjutkan ke proses mutasi. Proses mutasi ini bertujuan untuk menambahkan perbedaan dua vektor terhadap vektor ketiga dengan cara *random* untuk setiap vektor. Langkah awal dari proses mutasi ini adalah dengan membangkitkan nilai *random* vektor yaitu r0, r1, dan r2. Nilai r0, r1, dan r2 yaitu antara 0 sampai 9 karena sebanyak populasi vektor yang dimiliki. Setelah itu, akan dilakukan perhitungan menggunakan persamaan berikut.

$$V_{i,g} = x_{r0,g} + F(x_{r1,g} - x_{r2,g}) \quad (5)$$

Keterangan

$V_{i,g}$ = hasil vektor mutan ke i, pada generasi ke g

$x_{r0,g}$ = nomor *slot* pada vektor r0

$x_{r1,g}$ = nomor *slot* pada vektor r1

$x_{r2,g}$ = nomor *slot* pada vektor r2

F = parameter kontrol mutasi

Setelah dilakukan perhitungan, akan dilakukan pemeriksaan mesin yang memiliki persebaran nomor Perintah Kerja (PK) terbanyak. Nomor *slot* pada PK tertentu yang terletak di luar mesin dengan PK terbanyak akan diganti dengan nomor *slot* pada mesin dengan PK terbanyak.

Setelah itu, akan dilakukan proses *crossover*. Proses *crossover* ini bertujuan untuk menyilangkan setiap *slot* dari kode *order* yang terdapat pada vektor target dan vektor mutan. Langkah awal proses *crossover* yaitu dengan membangkitkan nilai *random* 0 sampai 1 sebanyak jumlah kode *order*. Apabila nilai *random* yang dibangkitkan lebih kecil dari nilai *crossover* (Cr), maka *slot* kode *order* pada vektor mutan akan dipilih sebagai *slot* kode *order* pada vektor hasil *crossover* atau disebut sebagai vektor trial. Sebaliknya, apabila nilai *random* yang dibangkitkan lebih besar dari nilai Cr, maka *slot* kode *order* pada vektor target akan dipilih sebagai hasil *crossover* atau vektor trial.

Setelah melalui proses *crossover*, akan dilakukan proses seleksi. Pada proses seleksi ini akan dihitung fungsi *fitness* dari setiap vektor target, vektor mutan, dan vektor trial. Fungsi *fitness* yang digunakan adalah untuk meminimasi waktu pengerjaan *order* dan *afal*. Hasil nilai *fitness* dari ketiga vektor tersebut kemudian akan diseleksi yaitu dengan mengambil 10 vektor terbaik yang memiliki nilai *fitness* minimum untuk digunakan sebagai vektor target pada generasi berikutnya. Fungsi *fitness* yang digunakan untuk menghitung waktu dan *afal* dapat dilihat pada persamaan berikut.

$$FungsiFitness = \sum P_i + \sum T_m \tag{6}$$

$$P_i = \frac{B_{jk}}{\sum B_{jk} \times j} + \frac{M_{jk}}{\sum M_{jk}} + V_{jk} \tag{7}$$

$$T_m = \frac{s_m \times n}{j \times t_m \times s_j \times n} \tag{8}$$

Keterangan:

P_i = nomor PK (Perintah Kerja) ke i

i = sebanyak nomor PK yang dijadwalkan

B_{jk} = bobot mesin ke j untuk meminimalkan persebaran nomor PK

M_{jk} = bobot mesin ke j untuk merapatkan persebaran kode *order* ke kiri

V_{jk} = variasi kemunculan nomor PK pada mesin ke j

n = waktu yang dibutuhkan mesin untuk mengerjakan 100kg plastik

k = maksimal kapasitas per mesin

T_m = nomor *order* ke m

m = sebanyak nomor *order* yang dijadwalkan

S_m = nomor *slot* yang paling besar pada *order* ke m

t_m = batas waktu pengerjaan untuk *order* ke m

s_j = jumlah slot dalam satu hari untuk mesin ke j

Fungsi untuk menghitung nilai *fitness* tersebut merupakan gabungan dari perhitungan waktu dan *afal*. Untuk menghitung *afal* digunakan beberapa gabungan fungsi yaitu pemberian bobot untuk setiap mesin agar persebaran nomor PK dapat seminimal mungkin, merapatkan persebaran kode *order* ke kiri, dan meminimalkan kemunculan yang bervariasi pada setiap

mesin. Hal ini diperhitungkan karena dapat meminimalkan *afal* yang dihasilkan. Bobot akan diberikan berdasarkan setiap kapasitas mesin sehingga untuk setiap ukuran mesin memiliki bobot yang berbeda-beda. Untuk nilai bobot masing-masing mesin dapat dibagi sebagai berikut:

1. Bobot persebaran nomor PK
   - Mesin kecil dimulai dari 28, 27, 26, 25,…, 1.
   - Mesin sedang dimulai dari 49, 48, 47, 46,…, 1.
   - Mesin besar dimulai dari 84, 83, 82, 81, 8,…, 1.
2. Bobot merapatkan kode *order*
   - Mesin kecil dimulai dari 1, 2, 3, 4,…, 28.
   - Mesin sedang dimulai dari 1, 2, 3, 4,…, 49.
   - Mesin besar dimulai dari 1, 2, 3, 4,…, 84.

4. Hasil dan Diskusi

4.1. Uji Coba Parameter Terbaik Algoritma *Differential Evolution*

Data pesanan yang digunakan untuk proses pengujian nilai *crossover rate* (Cr) terbaik adalah data pesanan / order pada Tabel 1. Data pesanan tersebut diuji dengan memberikan jumlah iterasi maksimum sebanyak 100 dan nilai Cr yang berbeda-beda dengan jumlah populasi yang digunakan adalah 10 vektor. Proses pengujian ini bertujuan untuk menentukan jumlah iterasi maksimum dan nilai Cr yang paling optimal sebagai nilai acuan untuk pengujian data pesanan lainnya.

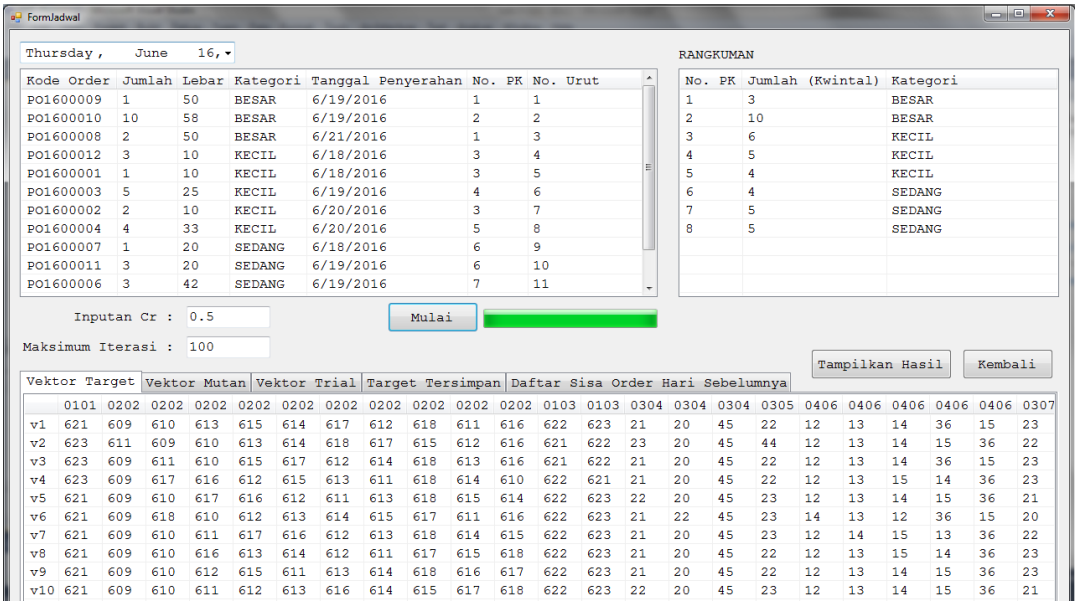
Berdasarkan data hasil pengujian apabila dilihat dari hasil penilaian dan waktu yang diperlukan aplikasi, maka nilai parameter Cr yang terbaik adalah 0,5. Jumlah iterasi yang paling efektif untuk digunakan adalah 100 karena rata-rata pada iterasi sebelum 100, aplikasi telah mendapatkan nilai yang paling optimal karena sudah tidak mengalami perubahan nilai *fitness*. Hasil dari pengujian parameter pada Tabel 9 digunakan sebagai nilai parameter *default* pada aplikasi dan sebagai acuan untuk melakukan pengujian data pesanan lainnya.

Tabel 9. Pengujian parameter DE

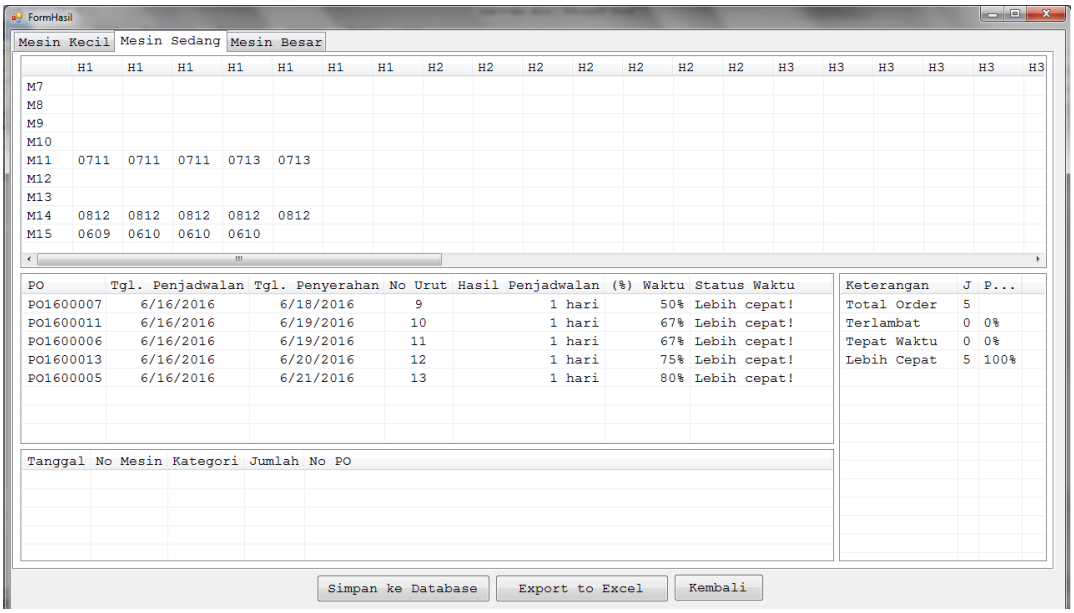
Uji Coba	Berhenti pada Iterasi ke-	Nilai Fitness Terbaik	Waktu
1	21	0,5288	56s
2	30	0,5185	81s
3	21	0,5288	54s
4	21	0,5288	58s
5	30	0,5185	86s
Rata-rata	24	0,5246	67s

4.2. Uji Coba Performa Penjadwalan

Pengujian ini dilakukan pada aplikasi untuk melihat performa penjadwalan yang dihasilkan oleh aplikasi. Uji coba dilakukan dengan membandingkan waktu pengerjaan, *afal* dan kecepatan penjadwalan yang diperoleh dari hasil perhitungan algoritma *Differential Evolution* terhadap metode penjadwalan yang sedang digunakan di lapangan. Hasil pengembangan aplikasi untuk inisialisasi dan penyebaran order pada slot mesin tiap hari setelah melewati proses DE dapat dilihat pada Gambar 3 dan Gambar 4.



Gambar 3. Inisialisasi Penjadwalan



Gambar 4. Penyebaran order dalam slot mesin

Data pesanan yang digunakan pada Tabel 1diuji dengan membandingkan penjadwalan secara manual dan penjadwalan dengan menggunakan aplikasi didapatkan hasil uji coba yang dapat dilihat pada Tabel 10.

Tabel 10. Perbandingan ujicoba penjadwalan manual dengan aplikasi

Kriteria	Hasil Penjadwalan Pesanan Tabel 1		
	Aplikasi	**Lapangan**	**% Efisiensi**
Makespan (jam)	227,2	230	1,21%
Afal (kg)	56	61	8,19%
Kecepatan (detik)	67	2700	97,51%

5. Kesimpulan dan Saran

Aplikasi untuk optimasi jadwal produksi dengan menggunakan algoritma *differential evolution* ini telah berhasil dan dapat menurunkan *makespan* sebesar 1,21%, *afal* sebesar 8,19%, dan waktu komputasi sebesar 97,51%

Referensi

[1] M. Astuti,"Studi Penjadwalan Jo Shop Untuk Meminimalkan Waktu Keseluruhan menggunakan Pendekatan Algoritma Artificial Immune System," *Angkasa.*, vol 5(1), pp. 19-28, 2013

[2] A. Sugioko,"Modifikasi Bee Colony Algorithm dengan Tabu List pada Penjadwalan Job Shop dengan Kriteria Biaya Keterlambatan,"Tesis,Universitas Indonesia,Jakarta, 2012

[3] D. D. Rohman and R. Ferdian," Penjadwalan 20 job 8 mesin dengan metode Genetic Algorithm (GA)," *Spektrum Industri*, Vol 11(2), pp. 175-184, 2013

[4] F. Amin,"Perancangan Aplikasi Penjadwalan Job Shop dengan Menggunakan Algoritma Simulated Anneling," *Jurnal pelita Informatika Budi Darma,*vol 8(3), pp. 62-68, Des.2014.

[5] S. E. Wiratno , R. Nurdiansyah, and B. Santoso,"Algoritma Differential Evolution untuk Penjadwalan Flow Shop Banyak Mesin Dengan Multi Obyektif," *Jurnal Teknik Industri*, Vol 13(1), pp. 1-6, Feb.2012

[6] M. Ali, M. Pant, and A. Abraham,"Simplex Differential Evolution," *Acta Polytechnica Hungarica*, vol 5(6), pp. 95-115,2009

[7] P. Bhaskara, G. Padmanabhan, and B. S. Kumar, 2015," Differential Evolution Algorithm for Flexible Job Shop Schedulling Problem," *Technical Research Organisation India*, 1(5), pp. 71-79, 2015

[8] I. A. Fajarwati and W. Anggraeni,"Penerapan Algoritma Differential Evolution untuk Penyelesaian Permasalahan Vehicle Routing Problem with Delivery and Pick-up," *Jurnal Teknik ITS,* Volume 1, pp. a391-a396, Sep.2012

[9] B. Qian, L. Wang, R. Hu, W. L. Wang, D. X. Huang, and X. Wang,"A Hybrid Differential Evolution Method for Permutation Flow-Shop Scheduling," *The International Journal of Advanced Manufacturing Technology*, Volume 38, pp. 757-777, Sep.2008

[10] Z. Čičková and S. Števo," Flow Shop Scheduling using Differential Evolution", *Management Information Systems*, vol 5(2), pp. 008-013, Aug.2010