BAB 2
KERANGKA TEORI DAN PEMODELAN

Bab ini berisi tentang kerangka teori yang dipergunakan untuk mendukung tesis ini. Beberapa bahasan meliputi teori penjadwalan, teori *job shop*, metode-metode yang digunakan untuk menyelesaikan permasalahan *job shop,* pemodelan masalah dan Algoritma *Differential Evolution* yang digunakan untuk menyelesaikan masalah tersebut.

2.1. Operasi Penjadwalan

Penjadwalan adalah suatu proses untuk memutuskan bagaimana menjalankan berbagai sumber daya pada berbagai tugas yang mungkin. Waktu dapat ditentukan atau dibuat mengambang sebagai bagian dari urutan kejadian[1].

Penjadwalan merupakan hal yang penting dalam organisasi untuk memeroleh pemanfaatan atau utilisasi yang optimal dari sumber daya produksi dan aset lain yang dimiliki. Penjadwalan diperlukan agar alokasi tenaga operator, mesin dan peralatan produksi, urutan proses, jenis produk, pembelian material dan sebagainya menjadi efisien. Di samping keputusan perencanaan jangka menengah yang tanpa memerhatikan urutan kegiatan produksi, ada masalah lain yang disebut penjadwalan yang mana alokasi sumberdaya dan urutan pengerjaan menjadi sangat penting. Dalam hierarki pengambilan keputusan, penjadwalan merupakan langkah terakhir sebelum dimulainya operasi.

2.1.1. Tipe-Tipe Proses Manufakturing dan Pendekatan Penjadwalan[2]

- Proses Kontinyu

Sistem proses kontinyu merupakan proses produksi yang cirinya adalah peralatan yang serba otomatis, penggunaan tenaga kerja rendah, fasilitas ditujukan untuk satu produk, seperti pembuatan produk kimia, *steel, wire*, dan kabel.

- Manufaktur dengan Volum Tinggi

[1] http://en.wikipedia.org/wiki/Scheduling (last updated 19 February 2010 at 12:58, accessed 01 April 2010

[2] Chase/Jacobs/Aquilano, *Operation Management*, Mcgraw-Hill, New York, 2007, hal. 667

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

Sistem ini memunyai karakteristik standarisasi peralatan atau kegiatan yang sama, contohnya pembuatan kendaraan bermotor, alat telepon, tekstil, peralatan rumah tangga atau kantor. Sistem ini cenderung mengacu pada *flow shop scheduling*, yang tujuannya untuk memeroleh arus barang yang lancar dalam rangka utilisasi tenaga kerja dan peralatan.

- Manufaktur dengan Volum Sedang

Sistem ini merupakan sistem produksi diantara volum tinggi dan volum rendah. Ini terjadi karena pesanan produksi belum cukup untuk dibuat masal; contoh produknya seperti produk konsumen *high-end* dan suku cadang industri.

- Manufaktur dengan Volum Rendah

Sistem ini disebut juga produksi *job shop* yang menghasilkan berbagai jenis produk yang sangat bervariasi, tetapi dibuat hanya dengan jumlah yang sedikit. Yang disampaikan dalam penelitian ini adalah di dalam kerangka penjadwalan sistem produksi *job shop*.

2.1.2. Tujuan Penjadwalan Pusat Kerja

Adapun tujuan penjadwalan pusat-pusat kerja sistem produksi *job shop* antara lain adalah [3]:

- Memenuhi tanggal jatuh tempo.
- Meminimumkan *lead time* yaitu meminimumkan periode waktu antara awal proses dan penyelesaian sebuah komponen.
- Meminimumkan waktu *set-up* (yaitu waktu yang dipergunakan sampai proses bisa dimulai) dan biaya.
- Meminimalkan persediaan (*inventory)* yaitu hasil pekerjaan yang masih dalam proses.
- Memaksimalkan pemanfaatan mesin, peralatan dan tenaga kerja.

Hal penting lainnya adalah menjamin bahwa tujuan pusat kerja sinkron dengan strategi operasional dari organisasi.

[3] Chase/Jacobs/Aquilano, *Operation Management*, Mcgraw-Hill, New York, 2007, hal. 667

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

2.1.3. Pembebanan

Pembebanan (*loading*) merupakan penugasan pekerjaan untuk memproses pekerjaan pada pusat-pusat kerja (*workstation)*, termasuk penugasan pekerjaan khusus pada pusat kerja. Ketika suatu tugas hanya dapat diproses pada pusat kerja tertentu, sedikit kendala akan dihadapi. Namun, bila terdapat dua atau lebih tugas yang akan diproses dan hanya terdapat sejumlah pusat kerja yang mampu mengerjakan tugas-tugas itu, maka akan timbul masalah yang kompleks. Pendekatan yang biasa dipakai untuk menangani masalah itu adalah *Gantt Chart* (yang diambil dari nama penemunya Henry Gantt 1910). *Gantt chart* berguna dalam pembebanan dan penjadwalan pada produksi dengan volume rendah. Bagan terdiri dari skala waktu yang digambarkan horisontal dan sumber daya pada arah vertikal.

Ada dua pendekatan yang berbeda yang dipakai untuk mengisi pusat-pusat kerja: *infinite loading* dan *finite loading*. *Infinite loading* menugaskan pekerjaan pada pusat-pusat kerja tanpa memperhitungkan kapasitas dari pusat-pusat kerja. Sedangkan dalam *finite loading* waktu mulai dan akhir dari masing-masing pusat kerja dan waktu proses pekerjaan diperhitungkan sedemikian rupa, sehingga tidak ada kapasitas berlebih.

2.1.4. Pengurutan Tugas (*Sequencing*)

Pengurutan tugas merupakan proses untuk menentukan urutan pekerjaan yang harus dikerjakan pada suatu pusat kerja, aturan prioritas mana dapat diklasifikasikan sebagai *local* atau *global.* Aturan lokal hanya berkenaan dengan satu stasiun kerja, sedangkan aturan *global* berkenaan dengan informasi banyak stasiun kerja. Aturan prioritas untuk urutan pekerjaan adalah sebagai berikut:

- FCFS (*first come first serve*): pekerjaan diproses berdasarkan pekerjaan yang datang lebih awal pada suatu pusat kerja.
- SPT (*shortest processing time*): pekerjaan diproses berdasarkan waktu proses yang paling singkat yang terlebih dahulu dikerjakan.
- EDD (*earliest due date*): pekerjaan berdasarkan *due date* setiap pekerjaan; jadi pekerjaan yang harus selesai paling awal dikerjakan lebih dulu.

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

- CR (*critical ratio*): pekerjaan yang memunyaI rasio paling kecil dari *due date* terhadap lama waktu proses dikerjakan terlebih dahulu.
- STR (*slack time to remaining*): waktu tersisa sebelum *due date* dikurangi waktu proses tersisa; pekerjaan yang memiliki *slack time* yang terkecil dikerjakan terlebih dahulu.
- *Rush:* keadaan darurat dimana pelanggan tertentu yang didahulukan.

2.1.5. Beberapa Definisi dalam Penjadwalan[4]

Berikut adalah beberapa pengertian yang berkaitan dengan penjadwalan mesin:

- Waktu proses *(processing time),* yaitu taksiran peramalan tentang berapa lama waktu yang dibutuhkan untuk menyelesaikan suatu tugas. Pada pembahasan ini, *processing time* dinyatakan dengan p_{ijk}.
- Waktu tenggat (*due date*), d_i, adalah batas waktu dari operasi terakhir dari suatu pekerjaan yang harus selesai. Untuk tugas yang terlambat penyelesaian *job* diluar waktu ini, maka akan dikenakan penalti pada *job* tersebut.
- *Slack time* adalah waktu tersisa yang muncul akibat dari waktu prosesnya lebih kecil daripada *due date*-nya.
- *Flow time* adalah rentang waktu antara satu titik pada saat tugas tersedia untuk diproses dengan suatu titik ketika tugas tersebut selesai.
- Waktu penyelesaian (*completion time*), C_{ij}, adalah waktu yang dibutuhkan untuk menyelesaikan pekerjaan mulai dari saat tersedianya pekerjaan (t=0) sampai selesai.
- *Makespan* biasanya dilambangkan dengan C_{max}, yaitu waktu pengerjaan seluruh *job.*
- Keterlambatan (*lateness*), $L_i = C_i - d_i$, adalah selisih antara waktu penyelesaian *job i* dengan waktu tenggatnya (*due date)*. *Lateness* baru dapat dihitung setelah *job i* selesai menjalani semua proses, dan dapat bernilai negatif, nol, atau positif.
- *Tardiness* adalah waktu terlambat yang bernilai positif jika suatu pekerjaan dapat diselesaikan lebih cepat dari *due-date*-nya.

[4] Bedworth Integrated Production Control System Management Analysis Design; Hal 299

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

- Heuristic adalah prosedur *rule of thumb* penyelesaian suatu masalah yang ditunjukkan untuk memproduksi hasil yang baik, tetapi tidak menjamin hasil yang optimal.

2.1.6. Karakteristik dan Kendala Proses

Berbagai karakteristik dan kendala penjadwalan produksi menurut Pinedo dan Chao (1999) dapat diklasifikasikan sebagai berikut:

- *Precedence constraint*

  Penjadwalan untuk setiap operasi dari job yang sama harus berurutan sesuai dengan *precedence constraint job* tersebut. Kendala ini terjadi ketika suatu *job* baru dapat mulai diproses setelah satu atau sekumpulan *job* lainnya selesai diproses
- *Sequence-dependent* terjadi ketika waktu *setup* mesin atau biaya untuk pekerjaan tertentu ditentukan, tidak hanya oleh pekerjaan itu, tetapi juga oleh pekerjaan sebelumnya.
- *Preemption*

  Preemption terjadi ketika suatu proses produksi sedang berlangsung, dapat disela atau dihentikan dan digantikan dengan mengerjakan *job* yang baru yang sifatnya lebih prioritas.
- Kendala sumber daya

  Kendala ini berkaitan dengan ketersediaan fasilitas produksi atau mesin yang akan digunakan untuk melaksanakan operasi pekerjaan yang ditugaskan sesuai dengan penjadwalan yang ditetapkan, dan juga berkaitan dengan ketersediaan tenaga kerja untuk menjalankan mesin-mesin tersebut.

2.1.7. Tujuan Penjadwalan

Tujuan dari aktifitas penjadwalan menurut *Bedworth* (1987) adalah sebagai berikut[5];

- Memenuhi *due date* pelanggan atau operasi hilir.
- Meminimumkan *flow time* (waktu penyelesaian sebuah pekerjaan).

[5] Chase/Jacobs/Aquilano, *Operation Management*, Mcgraw-Hill, New York, 2007, hal. 668

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

- Meminimumkan persediaan (*inventory*) WIP (*work in process*).
- Memaksimumkan utilisasi (minimasi waktu mesin dan pekerja yang menganggur).
- Meminimumkan keterlambatan baik *earliness* (penyelesaian lebih awal dari yang seharusnya) maupun *tardiness* (penyelesaian lebih lambat dari waktu yang ditentukan).
- Meminimumkan total biaya penalti atas keterlambatan.

2.2. Penjadwalan *Job Shop*

Job shop memiliki karakteristik produksi dengan sistem volum yang rendah, tetapi jenis produk yang dibuat bervariasi. Produksi dilakukan apabila ada pesanan dan biasanya sudah ditentukan kapan produksi yang bersangkutan diharapkan selesai. Apabila jumlah pesanan cukup banyak sedangkan fasilitas produksi terbatas, diperlukan penjadwalan dan pengurutan pesanan sebaik mungkin. Tujuan perusahaan dalam hal penjadwalan bisa berarti meminimalkan biaya penalti yang harus dibayar akibat keterlambatan, meminimalkan total waktu keseluruhan, meminimalkan total biaya produksi, dan lain-lain.

Penjadwalan pada proses *job shop* lebih sulit dibandingkan dengan penjadwalan *flow shop*. Hal ini dikarenakan alasan-alasan sebagai berikut:[6]

1. Variasi produk pada *job shop* sangat banyak, dengan pola aliran yang berbeda-beda melalui pusat-pusat kerja.
2. Peralatan pada *job shop* digunakan bersama-sama oleh bermacam-macam pesanan pada prosesnya, sedangkan pada *flow shop* hanya untuk satu produk.
3. *Job* yang berbeda mungkin ditentukan oleh prioritas berbeda pula. Hal ini mengakibatkan produk tertentu yang dipilih harus diproses seketika pada saat *order* tersebut ditugaskan pada pusat kerja. Sedangkan pada *flow shop* karena keseragaman *output*-nya tidak terjadi permasalahan tersebut.

Faktor-faktor tersebut menghasilkan sangat banyak kemungkinan kombinasi dari pembebanan (*loading*) dan urutan-urutan (*sequencing*). Contoh permasalahan *job*

[6] Rosnani Ginting, Penjadwalan Mesin, Graha Ilmu, 2009, p53

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

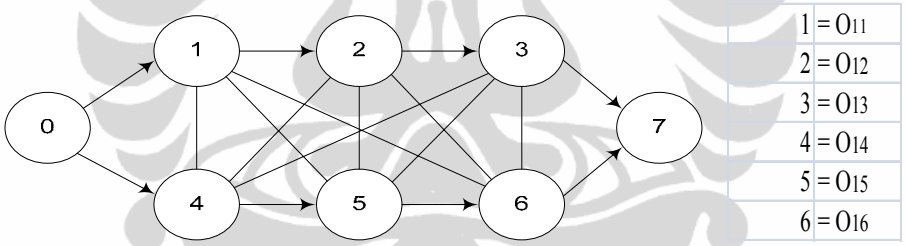
shop dapat digambarkan dengan *matrix* seperti yang ditunjukkan tabel berikut:

Tabel 2.1. Contoh 2x3 Permasalahan *Job Shop*

job	Mesin		
1	1 (10)	2 (15)	3 (20)
2	1 (5)	3 (30)	2 (8)

Pada contoh tabel 2.1 kita memunyai 2 *job*, 3 mesin dan urutan operasi yang ditunjukkan pada masing-masing baris dari *job*. Sebagai contoh *job* 1 yang diproses pada mesin 1 dengan waktu proses 10 menit, kemudian *job* 1 diproses pada mesin 2 dengan waktu 15 menit dan terakhir diselesaikan di mesin 3 dengan waktu 20 menit. Hal ini disebut urutan (*sequence*) dari *job* 1. Pada saat sebuah *job* *i* diproses pada sebuah mesin *j*, hal ini disebut sebagai "operasi (*i*,*j*)".

Bentuk permasalahan penjadwalan model *job shop* dapat digambarkan dalam gambar 2.1 berikut ini:



Gambar 2.1. Contoh Rute Penjadwalan *Job Shop*

Anak panah pada gambar melambangkan *precedence* setiap operasi. Operasi yang ada di sebelah kiri harus dikerjakan lebih dahulu dari operasi yang berada di kanan anak panah, jadi operasi O_{12} harus dikerjakan lebih dahulu, sebelum mengerjakan operasi O_{13}. Operasi 0 (awal) dan 7 (akhir) merupakan operasi semu yang ditambahkan untuk menyatakan bahwa kedua *job* tersebut berada pada satu masalah penjadwalan.

2.3. Metode-metode Penjadwalan Produksi

Ada beberapa metode untuk menyelesaikan masalah penjadwalan produksi, antara lain sebagai berikut:

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

2.3.1. Penjadwalan Heuristik

- Algoritma *Campbell, Dudek* dan *Smith* (CDS). Didasarkan pada algoritma *Johnson*, prinsipnya memecahkan masalah *n job* pada *m* mesin *flow shop* ke dalam *m-1* set persoalan dua mesin dengan membagi *m* mesin ke dalam dua grup, kemudian pengurutan *job* pada kedua mesin tadi menggunakan algoritma *Johnson.*
- Algoritma *Nawaz, Enscore* dan *Ham* (NEH). Metode ini juga disebut *Incremental Construction Algorithms,* diperkenalkan oleh *Nawaz, Enscore* dan *Ham* tahun 1983.
- Metode *Ignall-Scharge*, yang didasarkan pada prosedur percabangan (*branch*) dan pembatasan (*bound*). Percabangan membagi masalah yang besar dan rumit menjadi dua atau lebih sub masalah yang lebih sederhana, sedangkan pembatasan menghitung batas bawah solusi optimal dari sub masalah yang diperoleh dari percabangan.

2.3.2. Penjadwalan dengan Simulasi[7]

Simulasi penjadwalan pada intinya dilakukan dengan meninjau setiap jadwal operasi yang telah dibuat. Bila ditemukan suatu operasi dari *job* lain yang menggunakan mesin sama, tetapi dapat memberikan penghematan waktu tunggu, maka operasi dapat dibatalkan karena menyebabkan operasi lainnya harus menunggu lebih lama. Simulasi merupakan model tiruan dari sistem yang nyata, titik tolaknya adalah penyederhanaan sistem nyata dengan hanya memerhatikan beberapa bagian atau sifat utama yang memunyai hubungan sebab akibat dari sistem yang sebenarnya. Percobaan bisa dilakukan tanpa takut mengalami kegagalan dibandingkan dengan perlakuan pada kondisi nyata.

2.3.3. Penjadwalan *Heuristic* Modern

Algoritma *heuristic* modern atau yang lebih dikenal dengan *meta-heuristic* memecahkan masalah penjadwalan produksi dengan melakukan perbaikan mulai dengan satu atau lebih solusi awal. Solusi awal ini dapat dihasilkan secara acak, dapat pula dihasilkan berdasarkan *heuristic* tertentu. Empat algoritma *meta-*

[7] Ibid, p.205-206

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

heuristic yang dapat digunakan dalam memecahkan masalah penjadwalan *job shop* meliputi:

- Simulasi *Annealing*

Simulated annealing (SA) adalah salah satu algoritma untuk optimisasi yang bersifat generik. Berbasiskan probabilitas dan mekanika statistik, algoritma ini dapat digunakan untuk mencari pendekatan terhadap solusi optimum global dari suatu permasalahan. Masalah yang membutuhkan pendekatan SA adalah masalah-masalah optimisasi kombinatorial, di mana ruang pencarian solusi yang ada terlalu besar, sehingga hampir tidak mungkin ditemukan solusi eksak terhadap permasalahan itu. Publikasi tentang pendekatan ini pertama kali dilakukan oleh S. Kirkpatrick, C. D. Gelatt dan M. P. Vecchi (1983), diaplikasikan pada desain optimal *hardware* komputer, dan juga pada salah satu masalah klasik ilmu komputer yaitu *Traveling Salesman Problem*[8]. Prinsip dasarnya diambil dari bidang metalurgi, yang merupakan pemanasan materi di awal proses *annealing*, yang kemudian diikuti proses pendinginan yang perlahan-lahan memungkinkan atom-atom pada akhirnya menemukan tempat yang optimum.

- *Tabu Search (Gendrean, 2002)*

Dalam *Tabu Search*[9], kata taboo berasal dari *Tongan* yang digunakan oleh masyarakat aborigin di pulau Tonga untuk mengindikasikan barang-barang yang tidak dapat disentuh karena kesakralan benda tersebut. Kata tabu menyatakan memori masyarakat yang ditujukan pada perubahan setiap waktu. Metode TS didasarkan pada pemecahan masalah yang harus dikombinasikan dengan *adaptive memory* dan *responsive exploration. Adaptive memory* menyatakan bahwa TS menerapkan prosedur yang mampu mencari solusi secara efektif dan efisien. Ada hal yang berbeda dengan algoritma genetik dan simulasi *annealing*, keduanya tidak menggunakan memori dalam mengumpulkan informasi. *Responsive exploration* mengintegrasikan prinsip-prinsip dasar dari pencarian inteligen, untuk menemukan solusi yang optimal dengan menjelajahi area baru yang menjanjikan.

[8] http://www.en.wikipedia.org/wiki/Simulated_annealing, (last updated 4 Juni 2010, accessed 10 Juni 2010)

[9] Glover, Fred and Laguna Manuel. "Tabu search". Kluwer Academic Publishers. 1997. USA. Jurnal Internet : Hertz, Alain. " A Tutorial on Tabu Search". Canada.

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

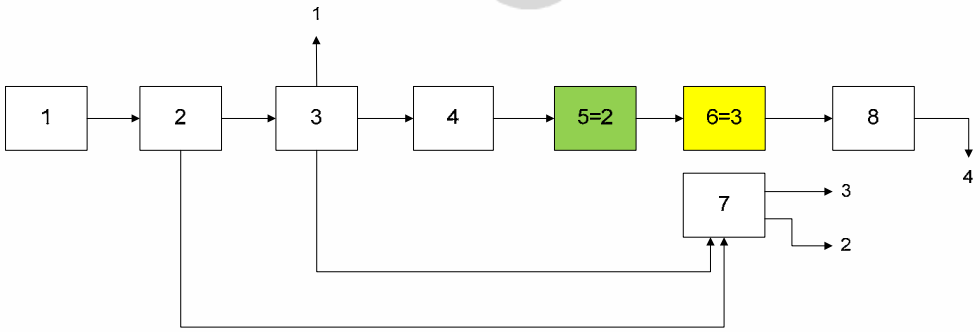
TS berkaitan dengan cara baru dan efektif untuk menemukan jalan dalam mengambil keuntungan dari mekanisme yang berhubungan dengan *adaptive memory* dan *responsive exploration*.[10]

- Algoritma Genetika

Metode Algoritma Genetika dikembangkan oleh John Holland (1970) dari *University of Michigan*. Metode ini termasuk salah satu metode yang bekerja berdasarkan pada prinsip seleksi alam dan teknik evolusi. Penelitian metodologi terhadap Algoritma Genetik terdiri atas penelitian dan aplikasi yang memerhatikan analogi-analogi ilmiah, analisa matematika dan perhitungan komputasi untuk mencari solusi atas masalah yang terjadi dari berbagai latar belakang studi. Algoritma Genetik mulai banyak digunakan untuk menyelesaikan berbagai masalah optimasi, karena langkah-langkahnya sangat sederhana dan tidak perlu mencari turunan fungsi untuk mencapai solusi optimum global.[11]

2.4. Pemodelan

Model penjadwalan *job shop* dalam penelitian ini dapat disebut dengan " *n job – m machine*". Jumlah *jobs* adalah n=10 yaitu *jobs* 1,2,3…10; dan jumlah mesin m=8, yaitu mesin 1,2,…5,6,…8. Mesin 5 dan mesin 6 merupakan mesin *dummy* dari mesin 2 dan mesin 3. Mesin 5 sama dengan mesin 2 dan mesin 6 sama dengan mesin 3, karena rute pengerjaan benda kerja mengharuskan kembali dikerjakan oleh mesin 2 dan mesin 3.



Gambar 2.2. Model Rute Operasi Produksi

[10] Rosnani Ginting, Penjadwalan Mesin, edisi pertama, Graha Ilmu, 2009, p.185.
[11] ibid, p.138.

Setelah semua *job* diidentifikasi rutenya masing-masing dirumuskanlah operasi *Wellhead* seperti gambar 2.2 dengan penjelasan pada tabel 2.2 sebagai berikut;

- *Job* 1 dan 2, rutenya adalah mesin 1, mesin 2 dan mesin 3.
- *Job* 3, 4 dan 5 rutenya adalah mesin 1, mesin 2 dan mesin 7.
- *Job* 6, 7 dan 8 rutenya adalah mesin 1, mesin 2, mesin 3 dan mesin 7
- *Job* 9 dan 10 rutenya adalah mesin 1, mesin 2, mesin 3, mesin 4, mesin 5, mesin 6 dan terakhir mesin 8.

Tabel 2.2. Urutan Operasi setiap *Job*

Job	**Nomor Mesin dan Urutan Operasi**							
	1	2	3	4	5	6	7	8
1 dan 2	1	2	3					
3,4 dan 5	1	2					7	
6,7 dan 8	1	2	3				7	
9 dan 10	1	2	3	4	5	6		8

Untuk menyelesaikan masalah diatas, digunakan pendekatan model Manne yang dimodifikasi oleh Cemal Ozguven et al.[12]. Model pada jurnal ini dipakai karena model ini dibangun dengan tujuan mengurutkan operasi-operasi pada mesin, sehingga waktu penyelesaian seluruh *job* menjadi minimum. Dari model Cemal Ozguven et al. ini yang dipakai adalah masalah fleksibilitas rute yang bersesuaian dengan permasalahan yang dihadapi dalam penelitian ini dengan tujuan meminimumkan total biaya produksi.

Pada penelitian Cemal Ozguven, model dibandingkan dengan model yang diajukan oleh Fattahi et al[13]. Untuk masalah dengan ukuran kecil kedua model saling bersinggungan; untuk ukuran sedang model Cemal Ozguven lebih baik dari model Fattahi. Namun, secara keseluruhan model Cemal Ozguven lebih baik dari model Fattahi dari sisi waktu CPU, nilai C_{max} dan ukuran model permasalahan. Model Cemal Ozguven dibandingkan dengan jurnal terbarunya Fattahi et al[14] memunyai kemiripan. Operasi dari *job* model Fattahi ini bisa *overlap* sepanjang

[12] Ozguven, Cemal. Et al., Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modeling* 34 (2010) 1539-1548.
[13] Fattahi, P. Et al. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Jurnal Intell. Manuf.* 18 (2007). 331-342.
[14] Fattahi, Parviz, et al. Flexible job shop scheduling with overlapping in operations. *Applied Mathematical modeling* 33 (2009). 3076-3087.

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

$C_i \geq \sum_k C_{ijk} \qquad \forall i \in J$

8. Waktu keseluruhan pekerjaan (*makespan*)

   $C_{max} \geq C_i \qquad \forall i \in J$

2.5. Algortima *Differential Evolution*

Algoritma ini dipakai sebagai metode untuk menyelesaikan model permasalahan *job shop* diatas.

2.5.1. Sejarah Algoritma DE

DE merupakan bagian dari algoritma evolusioner yang pertama kali di perkenalkan oleh Storn dan Price tahun 1995, yang kemudian dikembangkan menjadi fungsi optimasi yang handal dan serba guna yang bisa berlaku untuk berbagai masalah optimasi[15].

Ada juga sejumlah keunggulan yang signifikan bila menggunakan algoritma DE, yang diringkas oleh Price sebagai berikut[16]:

- Kemampuan untuk menemukan minimum global sebenarnya tanpa nilai-nilai parameter awal;
- Cepat dan sederhana dalam hal aplikasi dan modifikasi;
- Memerlukan beberapa parameter kontrol;
- Mampu memberikan solusi ganda dalam *run* tunggal;
- Kemampuan untuk menemukan solusi optimal untuk masalah optimisasi nonlinier dengan fungsi penalti.

Algoritma DE merupakan pengembangan dari *genetic algorithm* (GA), yaitu teknik pencarian solusi yang menirukan konsep evolusi natural melalui operasi reproduksi, pindah silang, dan mutasi. Perbedaan utama antara DE dan GA adalah pada skema mutasi DE yang *self adaptive* dan pada proses seleksi, semua solusi pada DE memiliki kesempatan yang sama untuk terpilih sebagai *parent*[17].

[15] K. V. Price, R. M. Storn and J. A. Lampinen, *Differential evolution: a practical approach to global optimization,* Springer, 2005.

[16] K. V. Price, "An introduction to differential evolution," in *New ideas in optimization,* D. Corne, M. Dorigo and F. Glover, Ed., McGraw-Hill, 1999

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

Pada tahun 1997, Storn dan Price telah membuktikan bahwa DE lebih akurat dan lebih efisien dibandingkan *simulated annealing* (metode berbasiskan probabilitas dan statistik) dan algoritma genetika (GA). Keunggulan DE dibandingkan algoritma yang lain adalah strukturnya yang sederhana, mudah untuk diaplikasikan, cepat, dan tangguh[18].

2.5.2.Konsep Dasar

Algoritma DE merupakan algoritma evolusi baru karena menerapkan variabel kode nyata dan biasanya bergantung pada mutasi sebagai operator pencarian. Kesamaan utama antara DE dengan algoritma Genetik adalah bahwa keduanya mempertahankan populasi dari populasi yang potensial dan menggunakan mekanisme seleksi untuk memilih individu terbaik dari populasi. Perbedaan utama antara DE dengan algoritma Genetik adalah sebagai berikut[19]:

- DE beroperasi secara langsung pada vektor *floating point* sementara CGA (*conventional genetic algorithm)* bergantung terutama pada string biner.
- CGA bergantung terutama pada rekombinasi untuk mengeksplorasi ruang pencarian, sedangkan DE menggunakan bentuk khusus dari mutasi sebagai operator dominan
- DE merupakan pemindahan evolusi di tingkat perilaku individu, menekankan hubungan perilaku antara individu dan keturunannya, sementara algoritma Genetik mempertahankan hubungan genetika.

Algoritma DE merupakan metode pencarian langsung paralel yang menggunakan populasi P dengan ukuran NP, yang terdiri dari calon-calon solusi. Populasi awal dibangun secara acak, sedangkan populasi berikutnya merupakan hasil evolusi dari individu-individu melalui iterasi yang disebut generasi.

---

[17] Dervis Karaboga and Selcuk Okdem, "A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm", *Turk J. Elec Engin*., vol. 12, no.1, 2004, p.53

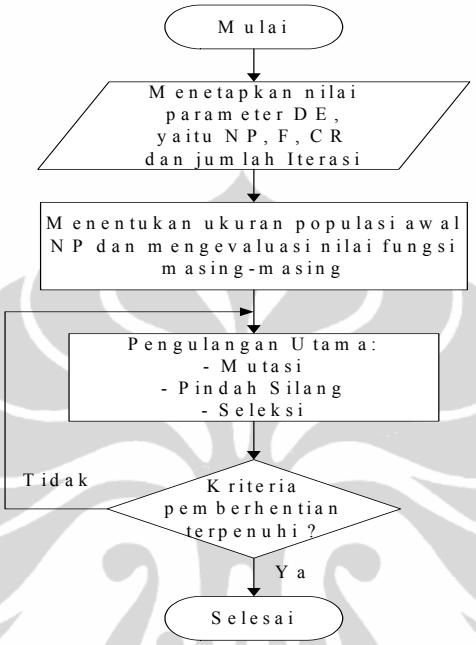
[18] Srikanta Routroy and Rambabu Kodali, "Differential Evolution Algorithm for Supply Chain Inventory Planning", in *Journal of Manufacturing Technology Management*, vol. 16, no.1, 2005, p.12.

[19] K. Price and R. Storn, "Differential evolution: a simple evolution strategy for fast optimization," *Dr. Dobb's Journal,* vol. 264, pp. 18-24, Apr. 1997

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

2.5.3. Proses Optimasi DE

Proses optimasi algoritma DE dapat dilihat pada gambar 2.3, yaitu sebagai berikut[20]:



Gambar 2.3. Diagram Alir Algoritma DE

- **Inisialisasi**

Pada tahap pertama optimasi DE, populasi calon solusi harus diinisialisasi. Secara khas masing-masing variabel keputusan dari masing-masing vektor pada populasi awal diberikan nilai bilangan acak. Dalam inisialisasi, sebuah populasi vektor NP, masing-masing dimensi D (jumlah variabel keputusan dalam masalah optimasi), secara acak menghasilkan lebih dari daerah layak. Nilai khas NP ini sekitar 5-10 kali D, untuk memastikan DE memunyai vektor yang cukup untuk bekerja[21]. Populasi awal (berisikan individu sejumlah NP) yang diinisialisasikan, merupakan solusi awal yang dapat diperoleh dari metode heuristik maupun diperoleh secara acak. Keistimewaan DE terletak pada langkah mutasi dua vektor yang dipilih secara acak dari populasi dan perbedaan vektor antara mereka disusun. Perbedaan ini dikalikan dengan suatu pembobotan faktor, F (ditentukan di awal dan tidak

[20] Neal M., et. al., "*Applying Differential Evolution to a Whole-Farm Model to Assist Optimal Strategic Decision Making*", 2006

[21] Storn, R. and Price, K. Differential Evolution - A simple and efficient heuristic for global ptimization over continuous spaces. *Journal of Global Optimization* 11 (4), pp. 341-359,1997

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

berubah selama algoritma) dan ditambahkan ke vektor ketiga yang dipilih secara acak dari populasi. Langkah ini dikenal sebagai variasi diferensial dan hasilnya dikenal sebagai vektor mutan (solusi). Nilai F adalah dalam kisaran 0 hingga 2. Algoritma DE sensitif terhadap pilihan F. Nilai F jarang lebih besar dari 1 karena akan menyebabkan DE mencari solusi di luar daerah layak. Demikian pula, F kurang dari 0,4 kurang efektif karena akan melibatkan DE mencari solusi yang mendekati daerah vektor target. Oleh karena itu pilihan awal yang baik untuk F akan menjadi 0,5.

- **Mutasi**

Proses optimasi DE dilakukan dengan menerapkan tiga dasar berikut genetik operasi; mutasi, rekombinasi (juga dikenal sebagai *crossover*) dan seleksi. Setelah populasi diinisialisasi, operator mutasi, *crossover* dan seleksi menciptakan generasi populasi P berikutnya (G +1) dengan menggunakan jumlah populasi P (G). Pada setiap generasi G, masing-masing vektor dalam populasi harus melayani satu kali sebagai vektor target Xi (G), vektor parameter memiliki indeks i, dan dibandingkan dengan vektor mutan. Operator mutasi menghasilkan vektor mutan (Vi (G)). Mutasi adalah proses pertukaran sejumlah gen dalam satu individu dengan menukar nilai karakter pada gen-gen tersebut dengan kebalikannya. Mutasi dilakukan untuk menjaga agar tidak terciptanya konvergensi prematur (solusi yang tidak optimal). Proses ini diformulasikan dengan rumus sebagai berikut:

$$V_i^{(G)} = X_{r1}^{(G)} + \mathrm{F}\ (X_{r2}^{(G)} - X_{r3}^{(G)}),\ \mathrm{i}=1,\ldots,\mathrm{NP} \qquad (2.1)$$

Penjelasan notasinya sebagai berikut:

$V_i^{(G)}$ = Vektor Mutan

F = Parameter kontrol mutasi

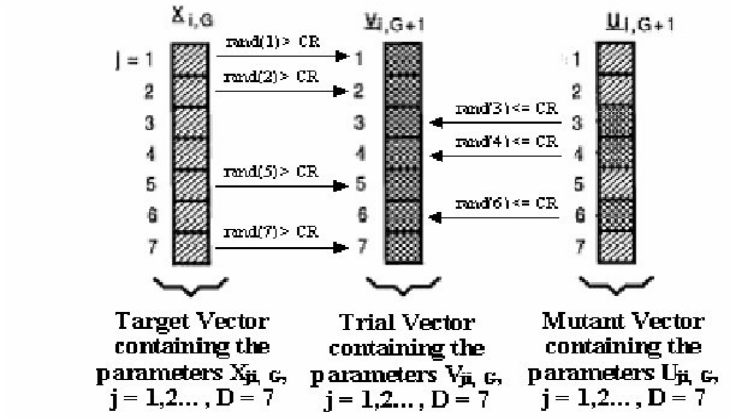
$X_{r1}^{(G)}, X_{r2}^{(G)}, X_{r3}^{(G)}$ = Vektor yang dipilih secara acak

- ***Crossover* atau Pindah Silang**

Crossover atau Pindah silang bertujuan untuk menambah keanekaragaman gen dalam populasi dengan penyilangan antar gen yang diperoleh dari produksi sebelumnya. Vektor mutan dikawinkan dengan vektor target X_i menggunakan

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

operasi pindah silang untuk menghasilkan vektor *trial*. Pada generasi G, *crossover* yang menciptakan vektor *trial* (U_i) dengan mencampur parameter dari vektor mutan (V_i) dengan vektor *target* (X_i) menurut sebuah distribusi probabilitas yang dipilih. Gen *trial* individual diwariskan dari V_i dan X_{r1} yang ditentukan melalui nilai faktor pindah silang (CR). Proses pindah silang akan dibantu dengan matriks baru yang nilainya diperoleh secara acak. Untuk kasus pada gambar 2.3, matriks baru merupakan matriks 7x1. Jika nilai baris matriks acak lebih besar dibandingkan dengan nilai CR, maka baris pada vektor target akan menjadi baris pada vektor *trial*, dan sebaliknya. Nilai CR berada pada kisaran [0,1].



Gambar 2.4. Proses Pindah Silang

Sumber: Roger, L.S. Tan et al.

- **Seleksi**

Penyeleksian dilakukan pada tahap akhir dari prosedur algoritma DE, untuk memilih individu manakah yang akan menjadi populasi generasi berikutnya (vektor target atau vektor *trial*). Vektor *trial* dapat menggantikan vektor target individual jika dan hanya jika nilai fungsi objektifnya lebih baik daripada nilai fungsi objektif vektor target. Proses optimasi DE terus berulang untuk meningkatkan kecocokan dari individu. Proses optimasi dihentikan setelah jumlah maksimum generasi tercapai atau kriteria konvergen tercapai.

- **Terminasi**

Proses pencarian akan berhenti jika telah dicapai kriteria terminasi. Namun, bila kriteria berhenti (terminasi) belum terpenuhi maka akan dibentuk lagi generasi baru dengan mengulangi langkah-langkah sebelumnya. Beberapa kriteria berhenti

yang sering digunakan antara lain: generasi/iterasi tertentu, waktu pencarian maksimum, dan nilai fungsi objektif (OBF) terbaik tidak lagi berubah.

DE memiliki beberapa varian[22], dinotasikan dalam *DE/x/y/z*, variabel *x* mendefinisikan vektor/individu yang akan dimutasi, bisa *random* ataupun *best vector*; *y* mendefinisikan jumlah *difference vector* yang digunakan; dan *z* mendefinisikan skema pindah silang yakni *binomial* atau *exponential*. Varian yang sering digunakan adalah DE/rand/1/bin[23]. Varian ini dianggap sebagai versi dasar dari DE.

2.6. Prosedur Optimasi Algoritma DE

Prosedur untuk melakukan optimasi dengan metode algoritma DE yang diharapkan dapat menyelesaikan masalah *job-shop* ditunjukkan dengan diagram alir yang tertera pada gambar 2.5.

Prosedur tersebut dijelaskan sebagai berikut[24]:

- Tahap 1. Inisialisasi
  - Menentukan parameter kontrol DE, yaitu ukuran populasi NP, faktor skala mutasi F, probabilitas pindah silang CR dan jumlah maksimum iterasi atau generasi.
  - Membangkitkan bilangan acak dan menempatkannya pada setiap vektor dari populasi awal.
  - Melakukan operasi permutasi (pengurutan), urutkan bilangan acak mulai dari yang terkecil sampai yang terbesar sesuai dengan aturan *Smallest Position Value* (SPV), disesuaikan dengan parameter keputusan dari masing-masing vektor.
  - Menghitung dan mengevaluasi setiap individu i dalam populasi dengan menggunakan fungsi objektif $f_i^0\left(\pi_i^0 \leftarrow X_i^0\right)$ (i = 1,2,…,NP), untuk memilih individu target.

[22] Nasimul Noman and Hitoshi Iba, "Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization", *GECCO'05*, Washington, DC, USA, 2005
[23] Hui-Yuan Fan, Jouni Lampinen, and Yeshayahou Levy, "An Easy-to-Implement Differential Evolution Approach for Multi-Objective Optimization", *in International Journal for Computer-Aided Engineering and Software*, 2006, vol. 23, no. 2, p.126
[24] *Ibid.*, p.8

Penjadwalan job..., Cecep Muntako, FT UI, 2010.

- Tahap 2. Memperbaharui generasi t=t+1
- Tahap 3. Menerapkan mutasi, pindah silang dan seleksi untuk mendapatkan individu baru;
  - Menerapkan operasi mutasi untuk mendapatkan vektor mutan, akan dicari individu mutan melalui operasi $V_i^G = X_{r1} + F\left(X_{r2}^G - X_{r3}^G\right)$, (r1 ≠ r2 ≠ r3≠i), dan r1, r2 dan r3 ϵ {1,...,Np}.
  - Menerapkan proses pindah silang untuk membentuk populasi *trial,* dengan mencampurkan parameter vektor mutan dengan vektor target sesuai dengan distribusi probabilitas terpilih.

$$u_i^{(G)} = u = \begin{cases} v_{j,i}^{(G)}, if & rand_j(0,1) \le CR \\ x_{j,i}^{(G)}, otherwise & \end{cases} \quad (2.2)$$

CR adalah konstanta pindah silang pada *range* (0,1), dan r_j adalah bilangan acak *uniform* antara 0 sampai 1.

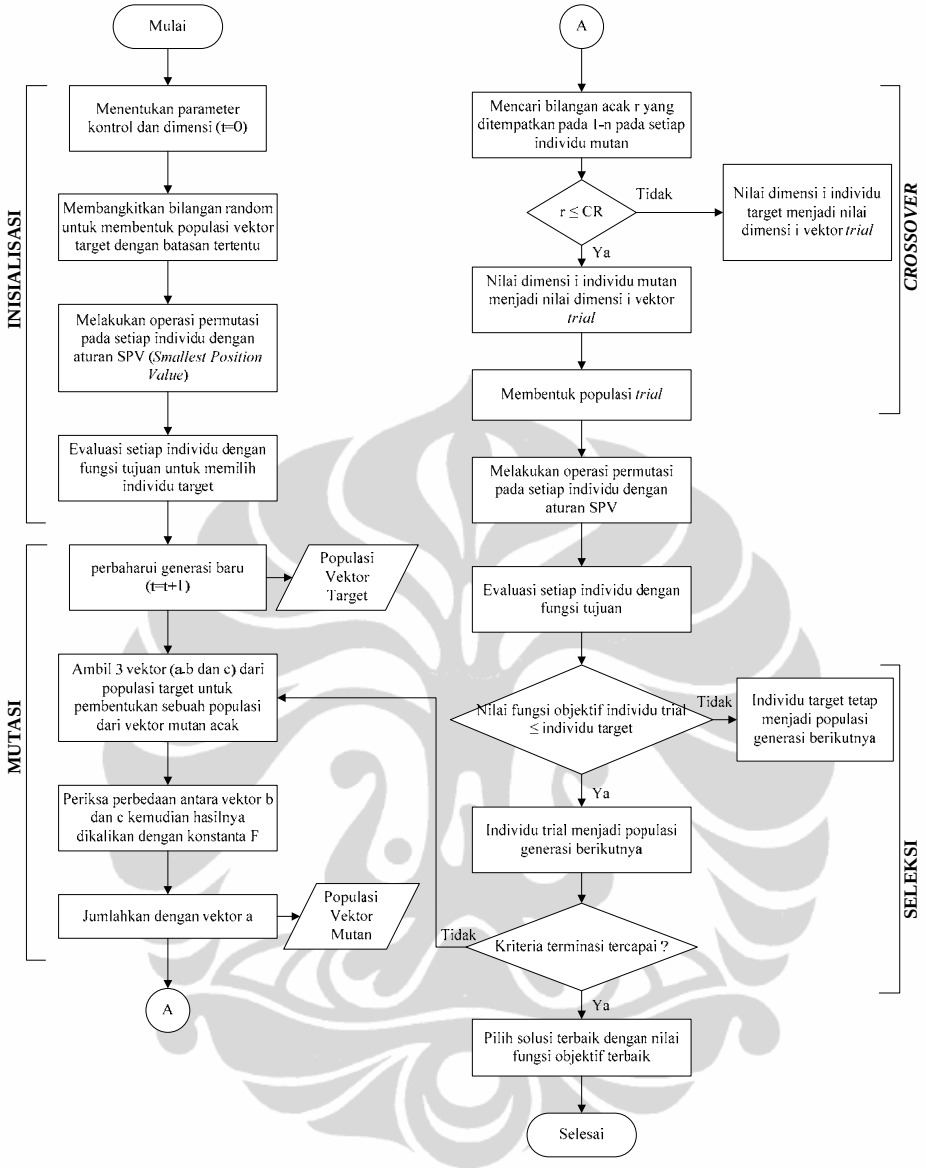
- Tahap 4. Melakukan operasi permutasi *job*

  Operasi permutasi job dilakukan dengan menerapkan aturan SPV untuk melakukan operasi permutasi $\varphi_i^t = \left[\varphi_{i1}^t, \varphi_{i2}^t, .., \varphi_{i,nm}^t\right]$ (i = 1,2,…,NP)
- Tahap 5. Mengevaluasi solusi terbaik dengan membandingkan populasi *trial* dan populasi target dan memilih solusi terbaik.
- Tahap 6. Melakukan penyeleksian.

  Nilai fungsi objektif individu *trial* U_i^{t+1} akan dibandingkan dengan individu target generasi sebelumnya, X_i^t, untuk menentukan apakah individu *trial* tersebut layak menjadi anggota populasi target generasi berikutnya atau tidak.
- Tahap 7. Memeriksa kriteria penghentian.

  Jika jumlah iterasi sudah mencapai jumlah iterasi yang telah ditentukan, maka algoritma akan berhenti, jika belum tercapai jumlah maksimum iterasi yang telah ditentukan, maka proses akan berulang mulai dari tahap 2.
- Tahap 8. *Output* terbaik merupakan hasil iterasi terakhir sebagai solusi terbaik terhadap permasalahan yang diteliti.

Penjadwalan job..., Cecep Muntako, FT UI, 2010.



Gambar 2.5. Diagram Alir Prosedur Optimasi DE