BAB II
DASAR TEORI

2.1 PENJADWALAN KERJA PADA PROSES PRODUKSI

Penjadwalan merupakan sebuah fungsi keputusan yaitu proses untuk menentukan sebuah jadwal (*schedule*). Dalam suatu proses produksi dalam arti luas, penjadwalan sangat dibutuhkan. Meskipun dalam hal ini disebut sebagai produksi, namun kegiatan didalamnya tidak hanya terbatas pada kegiatan manufaktur (industri) namun dapat juga dilakukan pada proses *supply chain*, logistik maupun jasa.

2.1.1 Pengertian Penjadwalan Produksi

Penjadwalan produksi adalah alokasi sumber daya dari waktu ke waktu untuk melaksanakan sekumpulan pekerjaan. Tujuan dari penjadwalan produksi adalah melakukan pengalokasian fasilitas produksi dalam hal ini mesin untuk melakukan suatu pekerjaan dengan menentukan urutan proses produksi suatu produk yang tepat agar dapat meminimalkan waktu pengerjaan produk (*makespan)* dan keterlambatan pesanan.

Menurut Everett dan Robert (1999), penjadwalan produksi merupakan bagian dari *shop floor control*, yang mencakup tahapan *loading*, *sequencing*, dan *detailed scheduling*. *Loading* ialah kegiatan dimana setiap *job* ditentukan rute prosesnya. Lalu, beban (*load*) yang harus diselesaikan oleh setiap mesin ditentukan dengan melihat *job* apa saja yang akan diproses oleh mesin tersebut. Pada tahap *sequencing,* ditentukan urutan pengerjaan setiap *job* baik secara keseluruhan atau untuk setiap mesin. Tahap terakhir yaitu *detailed scheduling,* yaitu tahap penentuan waktu mulai dan waktu selesai dari setiap operasi.

2.1.2 Jenis Penjadwalan Produksi

Penjadwalan secara garis besar dibedakan dalam penjadwalan untuk *flow shop* dan *job shop.* Perbedaan *flow shop* dan *job shop* adalah tidak adanya tahap-tahap proses yang sama dalam pola aliran kerja *jobshop*.

Penjadwalan *flow shop* adalah proses penentuan urutan pengerjaan yang memiliki lintasan produk yang sama. Model *flow shop* merupakan sebuah pekerjaan yang dianggap sbagai kumpulan dari operasi-operasi yang menerapkan sebuah struktur preseden khusus.

Pada model *flow shop,* operator dari suatu *job* hanya dapat bergerak satu arah yaitu proses dari awal sampai proses akhir. Diantara kedua proses tersebut tidak dimungkinakan untuk kembali ke proses sebelumnya.

Penjadwalan *job shop* adalah proses penentuan urutan pekerjaan untuk lintasan produk yang tidak beraturan. Secara umum, penjadwalan *job shop* dikenal dengan sekumpulan mesin dan sekumpulan pekerjaan yang akan dijawalkan.

Tipe pekerjaan yang dilakukan pada kegiatan *warehouse* ekspor, termasuk kedalam kegiatan penjadwalan *flow shop,* karena proses uuntuk lintasan produknya tida beraturan dengan jumlah mesin (dalam hal ini tim buruh) berbeda-beda.

2.2 PENJADWALAN *JOB SHOP*

2.2.1 Pengertian Penjadwalan *Job Shop*

Konsep penjadwalan *job shop* adalah menentukan waktu suatu produksi mulai dikerjakan dan mengalokasikan *resource*/mesin/tim pekerja untuk mengerjakan produksi tersebut. Pada saat menjadwalkan suatu produksi selain menentukan mulai dikerjakan juga menentukan *resource*/mesin/tim pekerja mana yang akan dipakai oleh operasi tersebut.

Job shop berisikan *m* jenis mesin yang berbeda, dimana masing-masing hanya dapat mengerjakan satu pekerjaan dalam satu waktu. Setiap kegiatan yang sudah berlangsung harus terus berjalan hingga selesai[1]. Sejumlah *n job* yang akan dikerjakan, dimasukkan kedalam daftar pekerjaan dengan ditentukan mesin manakah yang akan digunakan serta waktu yang diperlukan untuk menyelesaikan pekerjaan. Setiap pekerjaan dapat dilakukan oleh mesin yang berbeda.

Karakteristik *job shop* yang berbeda dengan *flow shop*, adalah dalam *job shop* dapat terjadi adanya perbedaan alur pekerjaan dengan berbagai macam jenis pekerjaan yang dilakukan[2].

---

[1] Jeffery, Barker, Graham McMahon, "Scheduling General Job Shop", *Journal of Informs Management Science*, Vol 31, No.5, 1985, hal 595

[2] Morton, Thomas, David Pentico, "Heuristic Scheduling Systems", *A Wiley-Interscience Publication*, 1993

Pada permasalahan *job shop*, *n* pekerjaan harus dikerjakan oleh m mesin dengan asumsi[3]:

- Ada sejumlah *m* mesin dan *n job*
- Setiap *job* terdiri dari satu rantai urutan operasi mengikuti rute yang telah ditentukan
- Setiap operasi dalam *job* diproses oleh salah satu mesin yang ada dengan waktu proses yang diasumsikan tetap
- Setiap operasi dapat melalui satu jenis mesin lebih dari satu kali (*recirculaton*)
- Tidak boleh ada *preemption* (penundaan satu *job* oleh *job* yang lain)
- Fungsi tujuan permasalahan penjadwalan model *job shop* adalah untuk mencari satu jadwal yang meminimalkan *makespan*
- Permasalahan penjadwalan *job shop* merupakan permasalahan optimisasi kombinatorial yang kompleks (*NP-hard*)

Secara garis besar prosedur umum yang diterapkan pada permasalahan penjadwalan dapat dibagi menjadi 2 kelompok, yaitu :

A. Constructive Procedure

Constructive procedure ialah suatu prosedur pemecahan permasalahan penjadwalan dimana solusi penjadwalan dibuat dalam satu kali proses pencarian sampai didapat satu solusi optimal yang lengkap. Metode yang termasuk kedalamnya antara lain :

- *Basic Dispatching Rules*
- *Mathematical Programming*
- *Composite Dispatching Rules*
- *Branch and Bound*
- *Beam Search*

B. Iterative Procedure

Iterative procedure berangkat dari satu solusi penjadwalan lengkap yang ditentukan secara acak atau dengan cara lain, yang kemudian solusi tersebut dimanipulasi secara bertahap untuk mendapatkan satu solusi yang optimal atau mendekati optimal.

[3] Charlier and Pinson, "An Algorithm for Solving The Job Shop Problem*", Journal of Informs Management Science*, Vol 32, No.2, 1989, hal 164

- *Classical Iterative Improvement*
- *Threshold Algorithms*
- *Tabu Search*
- *Simulated Annealing*
- *Genetic algorithms*
- *Algorithm Differential Evolution*

2.2.2 Masalah Penjadwalan *Job-Shop*

Masalah penjadwalan *job shop* merupakan persoalan pengurutan sejumlah operasi yang diproses pada mesin-mesin tertentu[4]. Masalah penjadwalan *job shop* adalah bagaimana menyusun semua operasi dari semua *job* pada tiap mesin dalam rangka meminimasi fungsi obyektif. Fungsi obyektif yang dimaksud dapat berupa waktu pengerjaan total, rata-rata waktu pengerjaan, rata-rata waktu keterlambatan penyelesaian job, atau lainnya[5].

Berdasarkan waktu kedatangan *job*, penjadwalan *job shop* dapat dikelompokkan sebagai *static job shop scheduling* dan *dynamic job shop scheduling*. Pada *static job shopscheduling*, semua *job* diterima pada saat yang sama. Pada *dynamic job shop scheduling*, waktu kedatangan job bervariasi tetapi sudah diketahui sebelumnya (*deterministic*) atau waktu kedatangan job bervariasi dan tidak dapat diketahui sebelumnya (*non deterministic*/ *stochastic*) [6].

2.3 ALGORITMA *DIFFERENTIAL EVOLUTION*

2.3.1 Definisi Algoritma *Differential Evolution*

Differential Evolution Algorithm merupakan algoritma optimasi global yang efisien, yang didasarkan pada prinsip evolusi[7]. Hampir sama seperti algoritma evolusi

---

4 Dimyati, T.T, dkk, 1999. "Model Optimasi untuk Integrasi Alokasi Produksi dengan Penjadwalan Operasi Job Shop dan Perencanaan Kapasitas", *Jurnal Teknik dan Manajemen Industri*, 19 (1), April 1999, 17-28.

5 Husbands, P., *Genetic Algorithms for Scheduling*, School of Cognitive and Computing Sciences, University of Sussex.

6 Lin, S., Goodman, E., Punch, W., 1997. *A Genetic algorithm approach to dynamic job shop scheduling problems*, dalam Back, T., editor, Proceedings of the Seventh International Conference on Genetic Algorithms, 481 – 489, Morgan Kaufmann.

7 K. V. Price, 1999, "*An Introduction to Differential Evolution",* dalam D. Corne, M. Dorigo, dan F. Glover, editors, *New Ideas in Optimization*, pages 79–108. Mc Graw-Hill, UK

lainnya, DE menggunakan individu sebagai representasi solusi kandidat. Setiap individu didefinisikan sebagai vektor berdimensi-d ($X \epsilon R^d$). Individu-individu tersebut merupakan anggota populasi pada generasi ke-g. Populasi dinotasikan sebagai $P(g)=\{X_a,X_b,\ldots,X_{NP}\}$. Notasi NP melambangkan ukuran populasi atau jumlah individu dalam populasi pada satu generasi. Nilai NP tidak berubah selama proses pencarian[8].

Algoritma ini mengeksploitasi populasi solusi potensial untuk menyelidiki ruang pencarian dengan mekanisme operasi mutasi, pindah silang sederhana, dan penyeleksian. Mutasi merupakan operasi utama yang memberi penekanan pada perbedaan sepasang individu acak anggota populasi[9]. Pindah silang menampilkan rekombinasi linear antara individu hasil mutasi (*mutant vector*) dengan satu orang tua (*target vector*) untuk menghasilkan satu anak (*trial vector*). Penyeleksian antara orang tua dengan anak bersifat deterministik (yang terbaik diantara keduanya akan menjadi anggota generasi berikutnya), dengan membandingkan fungsi objektif kedua individu yang bersaing tersebut. Proses ini berulang sampai dicapai titik optimum atau mendekati optimum, berdasarkan kriteria terminasi.

2.3.2 Tahapan Algoritma *Differential Evolution*

Tahap-tahap dalam algoritma DE meliputi inisialisasi, evalusi, mutasi, pindah silang, evaluasi, dan penyeleksian. Secara garis besar, algoritmanya dapat dilihat pada gambar 2.1 berikut ini.

```
Inisialisasi
Evaluasi
Ulangi
    Mutasi
    Pindah silang
    Evaluasi
    Penyeleksian
Sampai (ditemukan kriteria terminasi)
```

Gambar 2.1. Algoritma d*ifferential evolution*

---

[8] Lopez Cruz, L.G. Van Willigenburg and G. Van Straten, 2001, dalam *Proceedings of the IASTED International Conference Artificial Intelligence and Soft Computing*, May 21-24, Cancun, Mexico, pp. 211-216.

[9] Karaboga, Dervis and Selcuk Okdem, 2004, "A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm", Turk J Elec Engin, Vol.12.

(Sumber: Karaboga, 2004)

a) Inisialisasi

Tahap ini meliputi penetapan parameter kontrol dan penginisialisasian populasi awal. Penentuan parameter kontrol berdampak pada peforma DE (efektifitas, efisiensi, dan ketangguhan). Tujuan penentuan parameter kontrol adalah untuk menemukan solusi yang dapat diterima melalui sejumlah evaluasi fungsi. Tiga parameter kontrol dalam DE antara lain:

1). Parameter kontrol mutasi, F – faktor konstan dan real yang mengendalikan operasi mutasi, berada pada rentang [0,2].
2). Parameter kontrol pindah silang, CR – mengendalikan operasi pindah silang, berada pada rentang [0,1].
3). Ukuran populasi, NP – jumlah anggota populasi dalam satu generasi.

DE lebih sensitif terhadap pemilihan F daripada pemilihan CR[10]. CR berperan sebagai *fine tuning element* (elemen penentuan), pada saat operasi pindah silang. Nilai CR yang tinggi, misal CR=1, mempercepat terjadinya konvergensi. Terkadang, untuk beberapa permasalahan, nilai CR perlu diturunkan supaya DE lebih *robust* (tangguh). CR untuk DE yang menggunakan pindah silang binomial (misalnya tipe DE/rand/1/bin) biasanya lebih tinggi daripada untuk DE yang menggunakan pindah silang eksponensial (misalnya DE/rand/1/exp). Umumnya NP tidak berubah selama pencarian. Namun jika pencarian mengalami kondisi *stuck* maka NP dapat dinaikkan, atau dengan menaikkan F. F yang berada pada rentang [0.4,1] dinilai efektif.

Populasi awal yang diinisialisasikan merupakan populasi solusi awal. Solusi awal yang digunakan bisa diperoleh dari metode heuristik ataupun diperoleh secara acak. Populasi tersebut berisi individu sejumlah NP.

b) Evaluasi

Dari populasi yang ada tersebut, dilakukan evaluasi, untuk menyesuaikan nilai parameter individu terhadap nilai fungsi objektifnya. Pada saat ini dinilai juga individu mana yang layak dijadikan vektor target / individu target.

c)Mutasi

[10] http://www.aenf.wau.nl/mrs/staff/lopez/research/thesis/chap4.html

Individu yang tidak berperan sebagai individu target akan mengalami mutasi. Proses mutasi ini melibatkan beberapa individu (umumnya tiga). Proses mutasi diformulasikan dengan rumus:

$$X_{c'} = X_c + F(X_a - X_b)$$

d) Pindah silang

Dalam rangka mencapai keragaman yang lebih tinggi, individu mutasi $X_{c'}$ dikawinkan dengan X_d (individu target) menggunkan operasi pindah silang untuk menghasilkan keturunan atau individu *trial*. Gen individu *trial* diwariskan dari $X_{c'}$ dan X_d yang ditentukan melalui faktor pindah silang (CR). CR memberi aturan berapa banyak rata-rata gen yang bertalian dari individu mutasi dikopi ke keturunan.

e) Evaluasi

Individu *trial* akan dievaluasi, untuk menyesuaikan nilai parameter individu terhadap nilai fungsi objektifnya.

f) Penyeleksian

Terakhir, dilakukan proses penyeleksian, untuk memilih individu manakah (individu target ataukah individu *trial*) yang akan menjadi anggota populasi generasi berikutnya. Individu t*rial* dapat menggantikan posisi individu target pada generasi berikutnya jika dan hanya jika nilai fungsi objektifnya lebih baik daripada nilai fungsi objektif individu target.

g) Terminasi

Proses pencarian akan berhenti jika telah mencapai kriteria terminasi. Kriteria terminasi dapat ditentukan berdasarkan jumlah iterasi maksimum ataupun waktu proses.

DE memiliki beberapa varian[11], dinotasikan dalam *DE/x/y/z*, dimana *x* mendefinisikan vektor/individu yang akan dimutasi, bisa *random* ataupun *best vector*; *y*

[11] Noman, Nasimul *and* Hitoshi Iba, "*Enhancing Differential Evolution Performance with Local Search for High Dimensional Function Optimization*", *GECCO'05*, Washington, DC, USA, 2005

mendefinisikan jumlah *difference vector* yang digunakan; dan *z* mendefinisikan skema pindah silang yakni *binomial* atau *exponential*. Varian DE berikut ini berturut-turut adalah *DE/rand/1/exp, DE/best/1/exp*, *DE/rand/2/exp*, dan *DE/best/2/exp*:

$$y^i{}_{G+1} = x^j{}_G + \mathrm{F}\ (x^k{}_G - x^l{}_G) \quad (1)$$

$$y^i{}_{G+1} = x^{best}{}_G + \mathrm{F}\ (x^j{}_G - x^k{}_G) \quad (2)$$

$$y^i{}_{G+1} = x^j{}_G + \mathrm{F}\ (x^k{}_G - x^l{}_G) + \mathrm{F}\ (x^m{}_G - x^n{}_G) \quad (3)$$

$$y^i{}_{G+1} = x^{best}{}_G + \mathrm{F}\ (x^j{}_G - x^k{}_G) + \mathrm{F}\ (x^l{}_G - x^m{}_G) \quad (4)$$

Algoritma DE dapat diaplikasikan pada fungsi kontinyu *nonlinear*. DE memiliki beberapa keuntungan, antara lain konsepnya sederhana, cocok untuk aplikasi praktis, strukturnya sederhana, penggunaannya mudah, cepat dalam pencarian solusi, dan bersifat *robust* (tangguh, dalam arti memiliki standar deviasi yang kecil)[12].

2.3.3 Penerapan Algoritma DE pada Permasalahan Penjadwalan *Job Shop*

Algoritma DE yang akan diterapkan pada permasalahan ini menggunakan varian *DE/rand/1/bin* yang diperkenalkan oleh Storn dan Price[13] dengan SPV dalam algoritmanya. Adapun *pseudo code* algoritma DE yang akan digunkan adalah sebagai berikut [14]:

Inisialisasi parameter
Inisialisasi populasi target
Melakukan operasi permutasi
Evaluasi
Do{
    Membentuk populasi mutan
    Membentuk populasi *trial*
    Melakukan operasi permutasi

---

12 Routroy, Srikanta *and* Rambabu Kodali, 2005, "*Differential Evolution Algorithm for Supply Chain Inventory Planning*", *Journal of Manaufacturing Technology Management* vol.16 no.1

13 Tasgetiren, M. Fatih, et al, 2004, "Differential Evolution Algorithm for Permutation Flowshop Sequencing Problem with Makespan Criterion". Dalam Proceedings of the 4th International Symposium on Intelligent Manufacturing System (IMS2004), Sakarya, Turkey

14 Tasgetiren, M. F., Sevkli,M., Liang,Y.C., Yenisey, M.M, "Particle Swarm Optimization and Differential Evolution for Job Shop Scheduling Problem*", International Journal of Operational Research*, Vol. 3, No. 2, 2006, hal. 120-135.

Melakukan *Job Repetition*

Mengevaluasi populasi *trial*

Proses penyeleksian

}*While* (Berhenti)

2.3.3.1 Elemen Dasar Algoritma DE

Sebelum memulai proses pencarian solusi optimum pada permasalahan penjadwalan *job shop* dengan metode algoritma *Differential Evolution,* diperkenalkan terlebih dahulu elemen-elemen dasar algoritma DE yang akan digunakan. Elemen-elemen dasar tersebut adalah sebagai berikut[15]:

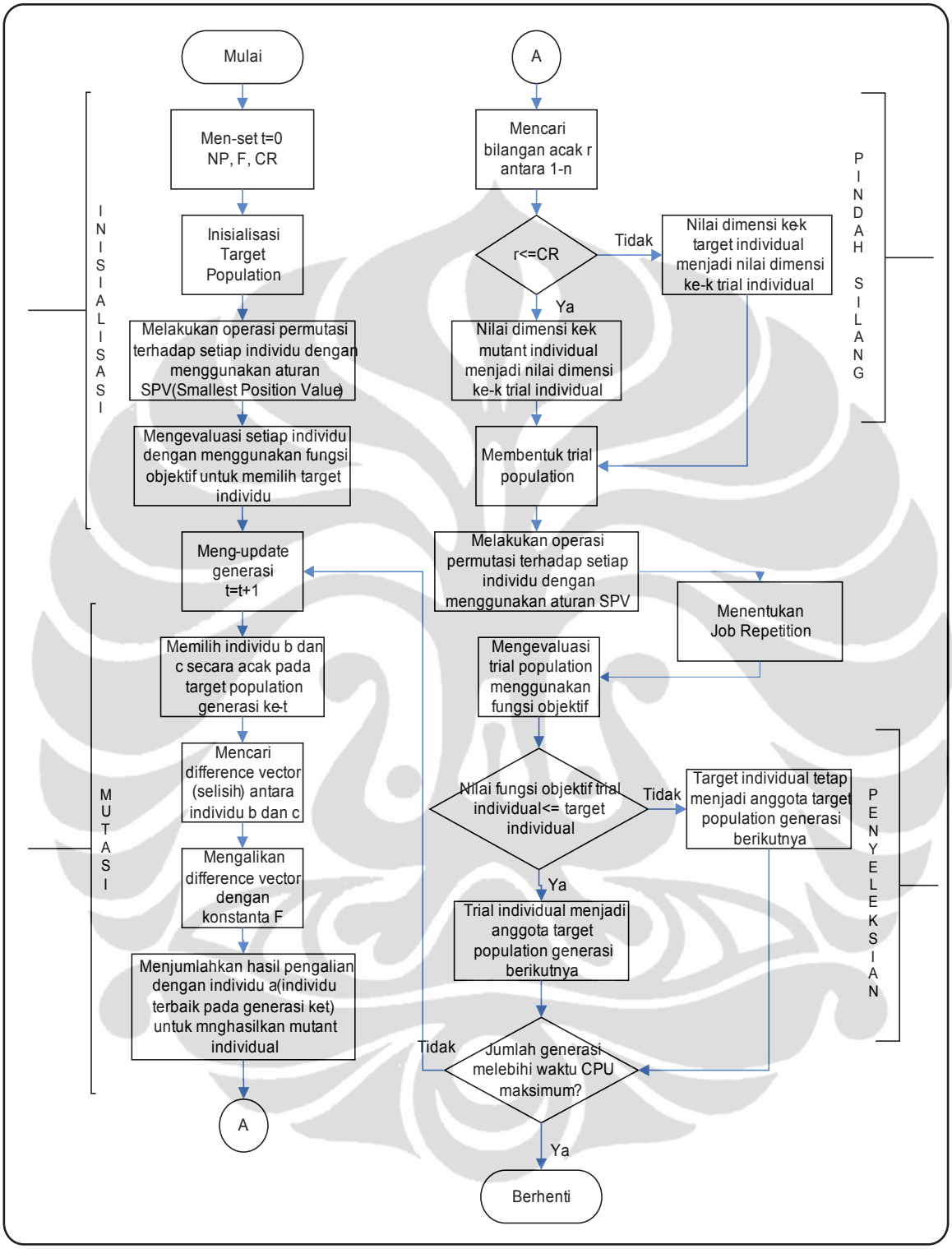
- **Individu target**: X_i^t individu ke-i anggota populasi pada generasi ke-t, dan diuraikan sebagai $X_i^t = \left|X_{i1}^t, X_{i2}^t, X_{i3}^t, \ldots, X_{nm}^t\right|$ dimana X_{ik}^t merupakan nilai dimensi individu ke-i terhadap dimensi ke-k (k = 1,2,...,nm).
- **Individu mutan**: V_i^t individu ke-i anggota populasi pada generasi ke-t, dan diuraikan sebagai $V_i^t = \left|V_{i1}^t, V_{i2}^t, V_{i3}^t, \ldots, V_{nm}^t\right|$, dimana V_{ik}^t merupakan nilai dimensi individu ke-i terhadap dimensi ke-k (k = 1,2,...,nm).
- **Individu *trial***: U_i^t individu ke-i anggota populasi pada generasi ke-t, dan diuraikan sebagai $U_i^t = \left|U_{i1}^t, U_{i2}^t, U_{i3}^t, \ldots, U_{nm}^t\right|$, dimana U_{ik}^t merupakan nilai dimensi individu ke-i terhadap dimensi ke-k (k = 1,2,...,nm).
- **Populasi target**: X^t sejumlah NP dalam *target population* pada generasi ke-t. Contoh : $X^t = \left|X_1^t, X_2^t, X_3^t, \ldots, X_{NP}^t\right|$
- **Populasi mutan**: V^t sejumlah NP dalam *target population* pada generasi ke-t**.** contoh : $V^t = \left|V_1^t, V_2^t, V_3^t, \ldots, V_{NP}^t\right|$
- **Populasi *trial***: U^t sejumlah NP dalam *target population* pada generasi ke-t. contoh : $U^t = \left|U_1^t, U_2^t, U_3^t, \ldots, U_{NP}^t\right|$

[15] Ibid

- **Operasi permutasi**: φ_i^t operasi permutasi *job* terhadap individu X_i^t. Diuraikan sebagai $\varphi_i^t = [\varphi_{i1}^t, \varphi_{12}^t, ..., \varphi_{i,nm}^t]$, dimana φ_{ik}^t merupakan penugasan operasi ke-j individu ke-i, pada iterasi ke-t.
- ***Job Repetition* :** Variabel π_i^t operasi pengulangan *job* terhadap individu X_i^t. Diuraikan sebagai $\pi_i^t = [\pi_{i1}^t, \pi_{12}^t, ..., \pi_{i,nm}^t]$, dimana π_{ik}^t merupakan penugasan operasi ke-j individu ke-i, pada iterasi ke-t. Dengan catatan Setiap penugasan dilakukan pengulangan n kali pada kromosom.
- **Konstanta mutasi**: F Є (0,2) merupakan konstanta angka *real* yang mempengaruhi variasi antara 2 individu
- **Konstanta pindah silang**: CR Є (0,1) merupakan konstanta angka *real* yang mempengaruhi besar populasi pada generasi selanjutnya
- **Fungsi objektif**: meminimumkan $f_i(\pi_i^t \leftarrow X_i^t)$ dimana π_i^t merupakan pengulangan dari individu X_i^t
- **Kriteria Terminasi**: adalah kondisi dimana program akan berhenti berjalan, bisa dalam bentuk jumlah maksimal generasi atau waktu maksimal CPU bekerja.

2.3.3.2 Prosedur Operasi Pencarian

Untuk melakukan proses pencarian solusi optimum pada permasalahan penjadwalan *job shop* dengan metode algoritma DE, dilakukan beberapa tahap atau prosedur. *Flowchart*-nya tertera pada gambar 2.2



Gambar 2.2. *Flowchart* algoritma DE untuk *job shop sequencing problem*

Prosedur tersebut dijabarkan sebagai berikut:

a) Tahap inisialisasi

- Menetapkan t = 0,

  NP = Jumlah dimensi

- Membangun individu acak sebanyak NP, yakni $\{X_i^t, i = 1,2,..., NP\}$, dimana

$$X_i^0 = [X_{i1}^0, X_{i2}^0, ..., X_{i,nm}^0];$$

- Melakukan operasi permutasi $\varphi_i^t = [\varphi_{i1}^t, \varphi_{12}^t, ..., \varphi_{i,nm}^t]$ untuk setiap individu $X_i^0 = [x_{i1}^0, x_{i2}^0, .., x_{i,nm}^0]$, i =1,2, ... , NP

- Menentukan Pengulangan $\pi_i^t = [\pi_{i1}^t, \pi_{12}^t, ..., \pi_{i,nm}^t]$ untuk setiap individu $X_i^0 = [x_{i1}^0, x_{i2}^0, .., x_{i,nm}^0]$, i =1,2, ... , NP

- Mengevaluasi setiap individu i dalam populasi menggunkan fungsi objektif $f_i^0(\pi_i^0 \leftarrow X_i^0)$ (i = 1,2,…,NP), untuk memilih individu target.

b) Meng-*update* generasi t = t + 1

c) Membentuk populasi mutan

- Untuk setiap individu target, X_i^t, i =1,2, ... , NP pada generasi ke-*t mutan* $V_i^{t+1} = [V_{i1}^{t+1}, V_{i1}^{t+1}, ..., V_{1,n}^{t+1}]$ yang diperoleh melalui operasi: $V_i^{t+1} = X_{a_i}^t + F(X_{b_i}^t - X_{c_i}^t)$, ($a_i \neq b_i \neq c_i$).

d) Membentuk populasi *trial*

- Operasi pindah silang dilakukan untuk meperoleh populasi *trial.* Untuk setiap individu mutasi, $V_i^{t+1} = [V_{i1}^{t+1}, V_{i2}^{t+1}, ..., V_{i,n}^{t+1}]$, merupakan angka integer acaka anatara 1 dan *nm,* $D_{i=(1,2,}$ ... , *nm*). *Individu trial* $U_i^{t+1} = [U_{i1}^{t+1}, U_{i2}^{t+1}, ..., U_{i,n}^{t+1}]$ diperoleh melalui operasi:

$$U_i^{t+1} = \begin{cases} v_i^{t+1}, if\ r_i^{t+1} \le CR\ or\ j = D \\ X_i^t, otherwise \end{cases}$$

D: dimensi acak (*k*=1,2,…,*nm*) yang menjamin setidaknya satu parameter setiap individu *trial* U_i^{t+1} berbeda dari generasi sebelumnya U_i^t.

CR: konstanta pindah silang pada *range* (0,1).

r_{ij}^{t+1} : bilangan acak *uniform* antara 0 sampai 1.

Individu *trial* di bentuk dari beberapa parameter individul mutasi, atau setidaknya salah satu parameter dipilih secara acak dari populasi target.

e) Melakukan operasi permutasi *job*

- Terapkan aturan SPV untuk melakukan operasi permutasi $\varphi_i^t = [\varphi_{i1}^t, \varphi_{12}^t, ..., \varphi_{i,nm}^t]$ (i = 1,2, …,NP).

f) Menentukan *job repetition*

- Menentukan pengulangan $\pi_i^t = [\pi_{i1}^t, \pi_{12}^t, ..., \pi_{i,nm}^t]$ (i = 1,2, …,NP).

g) Mengevaluasi populasi *trial*

- Evaluasi populasi *trial* menggunakan fungsi objektif $f_i^{t+1}(\pi_i^{t+1} \leftarrow U_i^{t+1})$ (i = 1,2,…,NP).

h) Melakukan penyeleksian

- Nilai *fitness* 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. Penentuan dilakukan dengan membandingkan populasi *trial* dan target populasi, yaitu :

$$X_i^{t+1} = \begin{matrix} U_i^{t+1}, if\ f(\pi_i^{t+1} \leftarrow U_i^{t+1}) \leq f(\pi_i^t \leftarrow X_i^t) \\ X_i^t, otherwise \end{matrix}$$

i) Memberhentikan operasi pencarian

- Jika jumlah generasi sudah melebihi waktu CPU maksimum, maka algoritma dihentikan, namun jika belum maka kembali ke tahap-2.