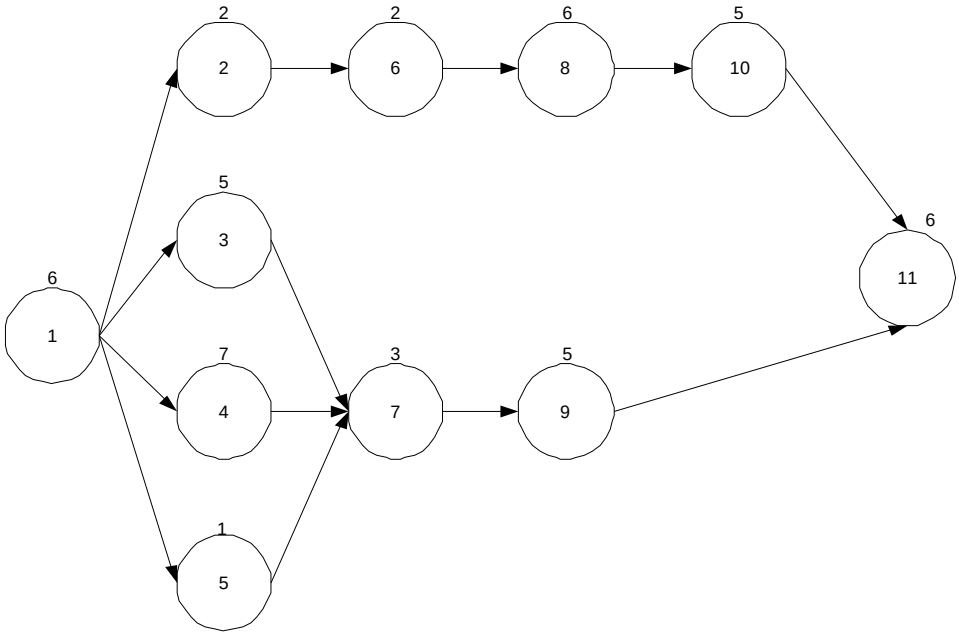
2. LANDASAN TEORI

2.1. Keseimbangan Lintasan

Menurut Wignojosoebroto (2000), keseimbangan lintasan merupakan suatu metode untuk mengendalikan atau menyeimbangkan lintasan produksi yang berkaitan dengan aspek waktu (waktu baku). Proses keseimbangan lintasan tidak dapat optimal karena waktu menganggur diperhitungkan di waktu baku sehingga dapat menyebabkan *idle*. Tetapi keuntungan dengan menerapkan keseimbangan lintasan adalah pengurangan aktivitas *material handling*, pembagian tugas yang merata, memacu operator untuk selalu bekerja. Tujuan menyeimbangkan lintasan adalah untuk meminimalkan waktu menganggur, meminimalkan jumlah stasiun kerja (Wignjosoebroto,2000).

Prosedur dalam menyeimbangkan lintasan adalah menempatkan elemen-elemen kerja pada tiap stasiun kerja dengan tidak melebihi waktu siklus. Dalam menempatkan elemen-elemen kerja pada tiap stasiun kerja yang perlu diperhatikan adalah *precedence constraint*. *Precedence constraint* merupakan hubungan suatu aktivitas untuk mendahului aktivitas lain yang dapat digambarkan dalam bentuk *precedence diagram*. *Precedence diagram* merupakan diagram sederhana sebagai prosedur dasar untuk mengalokasikan elemen-elemen aktivitas dimana memperlihatkan hubungan suatu aktivitas untuk mendahului aktivitas yang lain. *Precedence diagram* berfungsi untuk mempermudah penjelasan dari elemen-elemen aktivitas yang ditempatkan dalam suatu stasiun kerja. Contoh *precedence diagram* dapat dilihat pada Gambar 2.1.



Gambar 2.1. *Precedence diagram*

Sumber: Wignjosoebroto (2000, p.291)

Keterangan dari Gambar 2.1.:

1. Lingkaran-lingkaran bernomor menunjukkan elemen-elemen kegiatan dengan nilai waktu dicantumkan di luar lingkaran.
2. Arah panah menunjukkan hubungan antara satu kegiatan yang mendahului kegiatan lainnya.

Dari gambar di atas dapat diketahui bahwa elemen (3), (4), (5) akan dilaksanakan setelah elemen (1) selesai dilaksanakan. Tetapi elemen (3), (4), (5) tidaklah perlu harus menunggu elemen (2) selesai dikerjakan.

Setelah mengetahui hubungan elemen kerja maka perlu menetapkan waktu siklus, stasiun kerja minimum, dan lain-lain. Berikut merupakan rumus untuk waktu siklus, stasiun kerja minimum, efisiensi lintasan dan notasi-notasi yang digunakan (Wignjosoebroto,2000):

a. Cycle Time

Merupakan waktu siklus yang diperlukan oleh lintasan produksi untuk menghasilkan satu unit produk. Waktu siklus dapat ditentukan dari target produksi per periode yang hendak dicapai oleh perusahaan.

C = Jam kerja per periode / target produksi per periode (2.1)

Dimana:

C = waktu siklus

b. *Number of Work Station*

Setelah menentukan interval waktu siklus, maka jumlah stasiun kerja yang efisien dapat ditetapkan dengan persamaan:

$$m_{\min} = \frac{\sum_{i=1}^{n} t_i}{C} \qquad (2.2)$$

Dimana:

m_{min} = jumlah stasiun kerja minimum secara teoritis

n = jumlah elemen kerja

i = elemen kerja

t_i = waktu tiap elemen kerja i, dimana $i = 1, 2, 3, ..., n$

c. *Line Efficiency*

Mengukur keefisienan lintasan produksi akibat pengelompokkan elemen-elemen kerja ke stasiun-stasiun kerja. Apabila *line efficiency* besar berarti lintasan produksinya efisien.

$$E = \frac{\sum_{j=1}^{m_{\max}} ST_j}{K.C} \qquad (2.3)$$

Dimana:

j = stasiun kerja

ST_j = waktu stasiun kerja j, dimana $j=1, 2, 3, \ldots, m_{max}$

m_{max} = jumlah stasiun kerja maksimum yang diasumsikan sama dengan jumlah elemen kerja

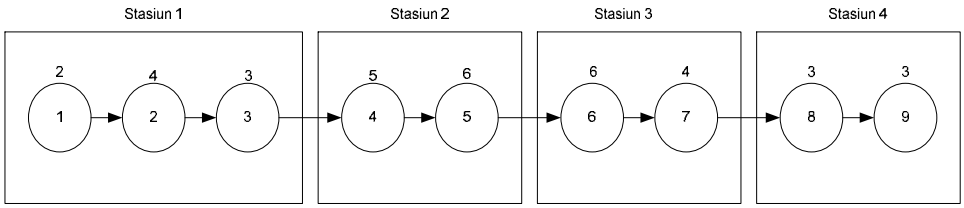
K = jumlah stasiun kerja

E = efisiensi lintasan

2.2. Straight Line Balancing

Menurut Elsayed dan Boucher (1994), *straight line balancing* adalah lintasan produksi dimana bahan baku dan bahan setengah jadi bergerak terus menerus dengan menempatkan elemen kerja pada stasiun kerja sehingga dapat

diketahui laju produksi. Lintasan untuk *straight line balancing* adalah garis lurus dapat dilihat pada Gambar 2.2.



Gambar 2.2. *Straight Line Balancing*

Sumber: Gokcen (2004, p.578)

Syarat untuk mencapai *straight line balancing* optimal dapat dilakukan dengan memperhatikan banyak elemen kerja yang dimiliki, *precedence constraint*, tempat yang tersedia untuk stasiun kerja, banyaknya elemen kerja. Masalah dalam *straight line balancing* adalah penempatan elemen kerja pada stasiun kerja, tetap mempertahankan *straight line balancing* pada lantai produksi (Elsayed dan Boucher, 1994).

Metode membuat *straight line balancing* adalah sebagai berikut:

- *Kilbridge-Wester*

  Setiap operasi kerja diberikan nomor untuk menggambarkan berapa banyak pendahulu yang dimiliki. Operasi kerja yang memiliki pendahulu terendah akan ditempatkan terlebih dahulu pada stasiun kerja. Langkah pengerjaan (Elsayed dan Boucher, 1994):

  1. Membangun *precedence diagram* untuk tiap elemen kerja. Pada *precedence diagram,* daftarkan kolom 1 semua elemen-elemen kerja yang tidak mengikuti satu sama lain. Kemudian di kolom 2, daftarkan elemen-elemen kerja yang mengikuti elemen-elemen kerja pada kolom 1, lanjutkan kolom berikutnya dengan cara yang sama.
  2. Menentukan waktu siklus (*C*).
  3. Tandai tiap elemen kerja pada stasiun dimana jumlah dari waktu elemen tidak melebihi *C*.
  4. Hapus elemen yang ditunjuk dari total nomor elemen kerja dan ulangi langkah ke 3.
  5. Jika waktu stasiun melebihi *C* maka elemen kerja tersebut harus dimasukkan ke stasiun berikutnya.

6. Ulangi langkah ke 3 hingga lengkah ke 5 sampai semua elemen terdaftar dalam stasiun kerja.

- *Helgeson-Birnie*

  Langkah-langkah dalam penggunaan metode *Helgeson-birnie* adalah sebagai berikut (Elsayed dan Boucher, 1994):

  1. Membuat *precedence diagram*
  2. Menentukan bobot posisi untuk tiap elemen kerja. Bobot posisi adalah lamanya waktu suatu elemen kerja dari elemen kerja awal hingga elemen kerja akhir yang mengikutinya.
  3. Meranking semua elemen kerja berdasarkan bobot posisi dari langkah 2. Elemen kerja dengan bobot posisi paling tinggi menempati posisi pertama.
  4. Menetapkan elemen kerja di stasiun kerja dimana elemen kerja yang memiliki bobot posisi tertinggi dan berdasarkan ranking akan ditempatkan pada stasiun pertama.
  5. Menetapkan elemen kerja selanjutnya ke stasiun kerja sesuai dengan *precedence diagram,* dan waktu stasiun kerja tidak boleh melebihi waktu siklus.
  6. Mengulang langkah 4 dan langkah 5 sampai semua elemen kerja berada di stasiun kerja.

- *Mixed Integer Programming*

  Notasi-notasi yang digunakan pada *straight line balancing* berdasarkan *linear programming* adalah sebagai berikut (Elsayed dan Boucher, 1994):

  x_{ij} = *1*, jika ada elemen kerja *i* berada di stasiun kerja *j*

  = *0*, jika tidak ada elemen kerja *i* berada di stasiun kerja *j*

  W_j = total elemen kerja yang berada pada stasiun kerja *j*

  Z_j = *1*, jika ada elemen kerja berada di stasiun kerja *j*

  = *0*, jika tidak ada elemen kerja berada di stasiun kerja *j*

  Berikut ini adalah fungsi tujuan dan kendala-kendala yang digunakan (Elsayed, 2004):

  - Fungsi Tujuan

    Meminimumkan jumlah stasiun kerja

$$Min \sum_{j=1}^{m_{\max}} z_j \tag{2.4}$$

- Kendala

a. Tiap elemen kerja berada pada satu stasiun kerja

$$\sum_{j=1}^{m_{\max}} x_{ij} = 1 \qquad \text{untuk } i = 1, \ldots, n \tag{2.5}$$

b. Waktu tiap stasiun kerja tidak melebihi waktu siklus

$$\sum_{i=1}^{n} t_i(x_{ij}) \le C \qquad \text{untuk } j = 1, \ldots, m_{max} \tag{2.6}$$

c. Memastikan jumlah stasiun kerja minimum

$$\sum_{i=\|W_j\|} x_{ij} \le \|W_j\| Z_j \qquad \text{untuk } j = 1, \ldots, m_{max} \tag{2.7}$$

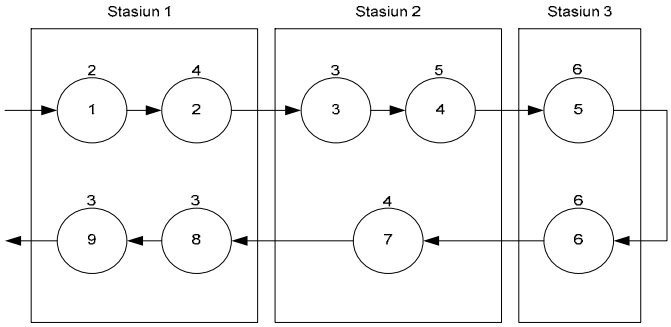
d. Hubungan antara elemen kerja dimana elemen kerja pendahulu berada di stasiun kerja terlebih dahulu daripada elemen kerja sesudah

$$\sum_{j=1}^{m_{\max}} [(m_{\max} - k + 1)(x_{rj} - x_{sj}) \ge 0 \quad \text{untuk } (r, s) \in P \tag{2.8}$$

2.3. U-*Line Balancing*

U-*line balancing* pertama kali dimunculkan oleh Miltenburg and Wijngaard pada tahun 1994 dengan menggunakan metode *dynamic programming*. Kemudian peneliti yang lain juga menggunakan prinsip U-*line balancing* tetapi dengan metode yang berbeda. Salah satu metode ini adalah *integer programming* yang diciptakan oleh Urban pada tahun 1998. Model U-*Line Balancing* dengan *integer programming* yang diteliti oleh Urban dapat menyelesaikan secara optimal (Urban, 1998).

U-*line balancing* memiliki lintasan yang berbentuk U dapat dilihat pada Gambar 2.3. Untuk U-*line balancing*, proses awal dan proses akhir statu produk dapat berada pada stasiun yang sama sehingga lebih fleksibel dan lebih efisien (Urban, 1998).



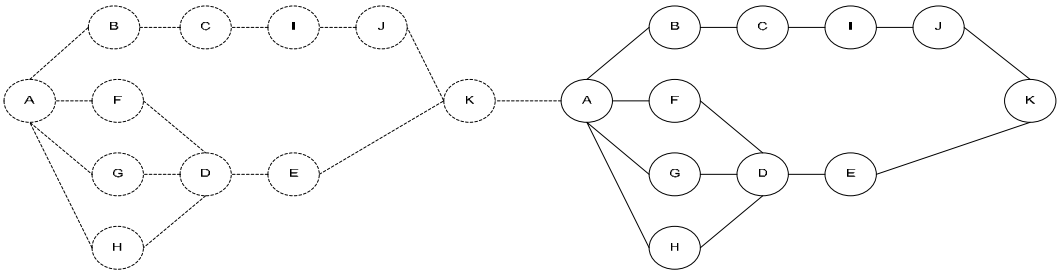
Gambar 2.3. U-*Line Balancing*

Sumber: Gokcen (2004, p. 578)

Tujuan dari U-*line balancing* adalah meminimalkan waktu siklus atau meminimalkan jumlah stasiun kerja, dimana meminimalkan waktu siklus akan meningkatkan laju atau kecepatan produksi sedangkan meminimalkan jumlah stasiun kerja akan menurunkan jumlah tenaga kerja (Urban, 1998).

Penyelesaian permasalahan U-*line balancing* dengan *integer programming* yang diciptakan oleh Urban dapat diselesaikan bila terdapat *n* elemen kerja, waktu tiap elemen kerja, t_i ($i = 1, \ldots, n$), dan *precedence relation* maka masalah yang terjadi yaitu penempatan elemen kerja di stasiun kerja ($j = 1, \ldots, m$) (Urban, 1998).

Urban (1998), mengusulkan untuk membuat *phantom network* dan menambahkan di *original precedence network*. Berikut adalah contoh *phantom network*:



Gambar 2.4. *Phantom Network*

Sumber: Urban (1998, p. 739)

Kelebihan dari U-*line balancing* adalah penempatan elemen-elemen kerja pada stasiun kerja dapat dibuat dari depan melalui *original network* dan belakang melalui *phantom network* atau secara bersamaan. Sehingga memungkinkan elemen kerja awal dan elemen kerja akhir pada satu stasiun kerja (Urban, 1998).

Notasi-notasi yang digunakan dalam pembuatan U-*line balancing* berdasarkan *integer programming* adalah sebagai berikut (Urban, 1998):

x_{ij} = *1*, jika elemen kerja *i* di *original precedence diagram* berada di stasiun kerja *j*; *i = 1, ..., n, j = 1, ..., m_{max}*

= *0*, jika tidak ada elemen kerja *i* di *original precedence diagram* berada di stasiun kerja *j*

y_{ij} = *1*, jika elemen kerja *i* di *phantom precedence diagram* berada di stasiun kerja *j*; *i = 1, ..., n, j = 1, ..., m_{max}*

= *0*, jika tidak ada elemen kerja *i* di *phantom precedence diagram* berada di stasiun kerja *j*

Z_j = *1*, jika ada elemen kerja berada di stasiun kerja *j*; *j = 1, ..., m_{max}*

= 0, jika tidak ada elemen kerja berada di stasiun kerja *j*

r = Elemen kerja pendahulu

s = Elemen kerja pengikut

P = *Precedence constraint*

Berikut *model integer programming* Urban:

- Fungsi tujuan:

  Meminimumkan jumlah stasiun kerja

$$Min \sum_{j=\lceil m_{\min} \rceil+1}^{m_{\max}} z_j \tag{2.9}$$

- Kendala:

  a. Semua elemen kerja harus berada pada stasiun kerja dan tiap elemen kerja hanya ditempatkan pada satu stasiun kerja.

  Rumus:

$$\sum_{j=1}^{m_{\max}} (x_{ij} + y_{ij}) = 1 \qquad \text{untuk } i = 1, ..., n \tag{2.10}$$

  b. Waktu proses di setiap stasiun kerja tidak boleh melebihi waktu siklus.

  Rumus:

$$\sum_{i=1}^{n} T_i (x_{ij} + y_{ij}) \le C \quad \text{untuk } j = 1, ... , \lceil m_{\min} \rceil \tag{2.11}$$

c. Waktu proses di setiap stasiun kerja setelah stasiun kerja ke-$\lceil m_{\min} \rceil$ tidak boleh melebihi waktu siklus jika stasiunnya ada.

Rumus:

$$\sum_{i=1}^{n} t_i (x_{ij} + y_{ij}) \leq C z_j \text{ untuk } j = \lceil m_{\min} \rceil + 1, ..., m_{max} \quad (2.12)$$

d. Hubungan tiap elemen kerja yaitu elemen kerja pengikut harus berada di stasiun berikutnya atau stasiun kerja yang sama dengan elemen kerja pendahulunya , baik untuk *original network* maupun *phantomed network.*

Rumus:

$$\sum_{j=1}^{m_{\max}} (m_{\max} - j + 1)(x_{rj} - x_{sj}) \geq 0 \quad \text{untuk } (r, s) \in P \quad (2.13)$$

$$\sum_{j=1}^{m_{\max}} (m_{\max} - j + 1)(y_{sj} - y_{rj}) \geq 0 \quad \text{untuk } (r, s) \in P \quad (2.14)$$