BAB II
LANDASAN TEORI

2.1 Programa linear

Programa Linear yang diterjemahkan dari *Linear programming (LP)* adalah suatu cara untuk menyelesaikan persoalan pengalokasian sumber-sumber yang terbatas di antara beberapa aktifitas yang bersaing, dengan cara yang terbaik yang mungkin dilakukan. Persoalan pengalokasian ini akan muncul manakala seseorang harus memilih tingkat aktifitas-aktifitas tertentu yang bersaing dalam hal penggunaan sumber daya langka yang dibutuhkan untuk melaksanakan aktifitas-aktifitas tersebut. Beberapa contoh situasi dari uraian di atas antara lain ialah persoalan pengalokasian fasilitas produksi, pengalokasian sumber daya nasional untuk kebutuhan domestik, penjadwalan produksi, solusi permainan (*game*), dan pemilihan pola pengiriman (*shopping*). Satu hal yang menjadi ciri situasi si atas ialah adanya keharusan untuk mengalokasiakan sumber terhadap aktifitas.

Programa linear ini menggunakan model matematis untuk menjelaskan persoalan yang dihadapinya. Sifat “linear” di sini memberi arti bahwa seluruh fungsi matematis dalam model ini merupakan fungsi yang linier, sedangkan kata “programa” merupakan sinonim untuk perencanaan. Dengan demikian, programa linear (LP) adalah perencanaan aktifitas-aktifitas untuk memperoleh suatu hasil yang optimum, yaitu suatuhasil yang mencapai tujuan diantara seluruh alternatif yang fisibel.

2.2 Tipe-tipe Khusus Persoalan Programa Linear

Programa Linear mempunyai beberapa karakteristik-karakteristik khusus, diantaranya ialah bahwa persoalan-persoalan tersebut cenderung membutuhkan sejumlah pembatas dan variabel yang sangat banyak sehingga penggunaan komputer dalam penyelesaian metodenya akan sangat mahal, atau mungkin proses perhitungannya akan menghadapi berbagai hambatan. Karakteristik lain ialah bahwa kebanyakan koefisiennya dalam pembatas-pembatasnya berharga nol, dan sedikit

sekali koefisien yang berharga bukan nol terjadi atau muncul dalam suatu pola tertentu. Karena itu, penting bagi kita untuk mengenal tipe-tipe khusus dari persoalan ini sehingga, jika pada suatu saat persoalan ini muncul, kita akan segera mengenal dan menyelesaikan dengan prosedur perhitungan yang tepat.

Tipe khusus persoalan programa linear yang paling penting ialah apa yang dikenal sebagai persoalan transportasi. Selain itu juga ada yang dikenal dengan persoalan *transshipment* dan persoalan penugasan *(Assignment)* yang erat kaitannya dengan persoalan transportasi.

2.2.1 Persoalan Transportasi

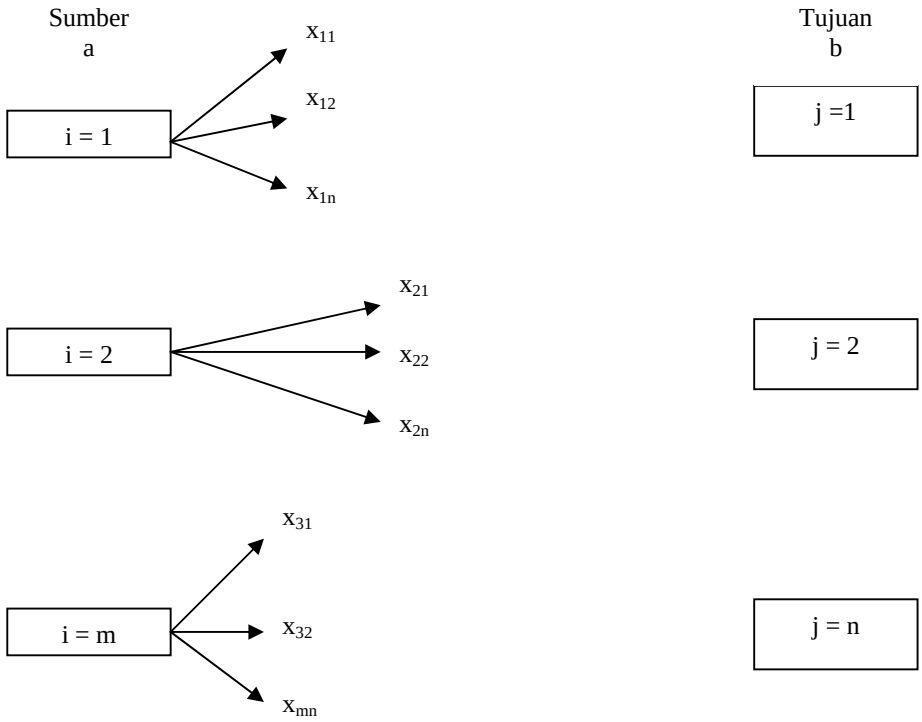
Persoalan transportasi membahas masalah pendistribusian suatu komoditas atau produk dari sejumlah sumber *(supply)* kepada sejumlah tujuan *(destination, demand)*m dengan tujuan meminimumkan ongkos pengangkutan yang terjadi.

Ciri-ciri khusus persoalan transportasi ini adalah :

1. Terdapat sjumlah sumber dan sejumlah tujuan tertentu.
2. Kuantitas komoditas atau barang yang didistribusikan dari setiap sumber dan yang diminta oleh setiap sumber dan yang diminta oleh setiap tujuan,besarnya tertentu.
3. Komoditas yang dikirim atau diangkut dari suatu sumber ke suatu tujuan, besarnya sesuai bengan permintaan dan atau kapasitas sumber.
4. Ongkos pengangkutan komoditas dari suatu sumber ke suatu tujuan, besarnya tertentu.

2.2.1.1 Model Transportasi

Secara diagramatik, model transportasi dapat digambarkan sebagai berikut :
Misalkan ada m buah sumber dan n buah tujuan.



Gambar 2.1 Model Transportasi

→ Masing-masing sumber mempunyai kapasitas a_i, i = 1, 2, 3, ..., m

→ Masing-masing tujuan membutuhkan komoditas sebanyak b_j, j = 1, 2, 3, ..., n

→ Jumlah satuan (unit) yang dikirimkan dari sumber i ke tujuan j adalah sebanyak x_{ij}

→ Ongkos pengiriman per unit dari sumber i ke tujuan j adalah c_{ij}

$$\text{Minimumkan}: z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij}$$

$$\text{Berdasarkan Pembatas}: \sum_{j=1}^{n} x_{ij} = a_i,\ i = 1, 2, 3, ..., m$$

$$\sum_{i=1}^{m} x_{ij} = b_j,\ j = 1, 2, 3, ..., n$$

$$x_{ij} \geq 0 \text{ untuk seluruh i dan } j$$

2.2.2 Model *Transshipment*

Model *Transshipment* adalah model transportasi yang memungkinkan pengiriman barang (komoditas) secara tidak langsung, dimana barang dari suatu sumber dapat berada pada sumber lain atau tujuan lain sebelum mencapai tujuan akhirnya. Jadi, pada Model *Transshipment* ini, suatu simber sekaligus dapat berperan sebagai tujuan dan sebaliknya, suatu tujuan dapat juga berperan sebagai sumber.

Dalam model ini, setiap sumber maupun tujuan dipandang sebagai titik-titik potensial bagi *demand* maupun *supply*. Oleh karena itu, untuk menjamin bahwa tiap titik potensial tersebut mampu menampung total barang di samping jumlah barang yang telah ada pada titik-titik tersebut, maka perlu ditambahkan kepada titik-titik itu kuantitas *supply* dan *demand*-nya masing-masing sebesar B.

$$B \geq \sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$$

2.2.3 Model Penugasan

Model penugasan merupakan kasus khusus dari model transportasi, dimana sejumlah m sumber ditugaskan kepada sejumlah n sumber situgaskan kepada sejumlah n tujuan (satu sumber untuk satu tujuan) sedemikian sehingga didapat ongkos total yang minimum.

Biasanya yang dimaksud dengan sumber ialah pekerjaan (*job*), sedangkan yang dimaksud dengan tujuan ialah mesin-mesin (pekerja). Jadi, dalam hal ini, ada m pekerjaan yang ditugaskan pada n mesin, dimana apabila pekerjaan i (i = 1, 2, 3, ..., m) di tugaskan kepada mesin j (j = 1, 2, 3, ..., n) akan muncul ongkos penugasan c_{ij}. Karena satu pekerjaan ditugaskan ditugaskan hanya pada satu mesin, maka *supply* yang dapat digunakan pada setiap sumber adalah 1 (atau $a_i = 1$, untuk seluruh i). Semikian pula halnya dengan mesin-mesin, karena satu mesin hanya dapat menerima satu pekerjaan, maka *demand* dari setiap tujuan adalah 1 (atau $b_j = 1$, untuk seluruh j). Jiks ada suatu pekerjaan yang tidak dapat ditugaskan pada mesin tertentu, maka c_{ij}

yang berkorespondensi dengannya dinyatakan sebagai M, yang merupakan ongkos yang sangat tinggi.

Penggambaran umum persoalan penugasan ini adalah sebagai berikut :

	1	2	...	N	
1	c_{11}	c_{12}	...	c_{1n}	1
2	c_{21}	c_{22}	...	c_{2n}	1
.	.	.		.	.
.	.	.		.	.
.	.	.		.	.
M	c_{m1}	c_{m2}	...	c_{mn}	1
	1	1	...	1	

Gambar 2.2 Gambaran Umum Persoalan Penugasan

Sebelum model ini dapat dipecahkan dengan sebuah metode, terlebih dahulu persoalannya harus diseimbangkan dengan menambahkan pekerjaan-pekerjaan atau mesin-mesin khayalan, bergantung pada apakah m < n atau m > n. Dengan kata lain nilai m harus sama dengan n agar semua mesin-mesin mendapatkan pekerjaan masing-masing satu.

Secara matematis, model penugasan ini dapat dinyatakan sebagai berikut :

$$Xij \begin{cases} 0, \text{ jika pekerjaan ke - i tidak ditugaskan pada mesin ke - j} \\ 1, \text{ jika pekerjaan ke - i ditugaskan pada mesin ke - j} \end{cases}$$

Dengan demikian, model persoalan penugasan ini adalah ;

$$\text{Minimumkan} : z = \sum_{j=1}^{n} c_{ij}\, x_{ij}$$

Berdasarkan Pembatas:

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, 2, ..., n$$

$$\sum_{i=j}^{n} x_{ij} = 1, j = 1, 2, ..., n$$

$$x_{ij} = 0 \text{ atau } 1$$

Suatu ciri khas persoalan penugasan ialah bahwa solusi optimum akan tetap sama bila suatu konstanta ditambahkan atau dikurangkan kepada baris atau kolom yang manapun dari matriks ongkosnya.hal ini dapat dibuktikan sebagai berikut :

Jika pi dan qj meripakan konstanta pengurang terhadap baris i dan kolom j, maka elemen ongkos yang baru adalah :

$$cij' = c_{ij} - p_i - q_j$$

Sehingga fungsi tujuan baru menjadi :

$$z' = \sum_i \sum_j c_{ij} \qquad xij = \sum_i \sum_j (c_{ij} - p_i - q_i) x_{ij}$$

$$= \sum_i \sum_j c_{ij}\, x_{ij} \quad - \quad \sum_i p_i \sum_j x_{ij} \; - \; \sum_i q_1 \sum_j x_{ij}$$

$$= \text{Karena } \sum_j xij \quad = \quad \sum_i xij = 1, \text{ maka } z' = z - \text{konstanta}$$

Hal ini menunjukan bahwa meminimumkan z menghasilkan solusi yang sama dengan meminimumkan z’.

Suatu hal yang menarik ialah bahwa jika kita melakukan operasi pengurangan p_i dan q_j tehadap matriks ongkos akan diperoleh *zero enteries*, yaitu elemen-elemen ongkos dalam matriks yang berharga nol, yaitu juga merupakan variable-variabel yang menghasilkan solusi optimum bagi z’ sehingga, berdasarkan pembuktian di atas, merupakan solusi optimal bagi z.

2.3 *Quadratic Assignment Problem (QAP)*

Quadratic Assignment Problem (QAP) pertama kali dikenalkan oleh Koopmans dan Beckman tahun 1947, yang menyatakan bahwa *Quadratic Assignment Problem (QAP)* adalah suatu model untuk menguraikan permasalahan yang praktis.

Permasalahan *Quadratic Assignment Problem (QAP)* adalah suatu masalah optimasi kombinatorial dan termasuk dalam permasalahan penugasan yang bertipe *linear programming*. Permasalahan penugasan *Quadratic Assignment Problem (QAP)* akan mencakup sejumlah n sumber yang mempunyai n tugas. Dengan matriks berukurab n x n.

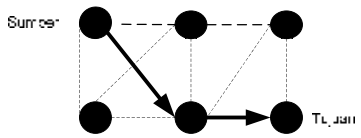
Tiap penugasan mempunyai urutan sumber untuk urutan tugas masing-masing sumber. Optimasi dari permasalahan penugasan *Quadratic Assignment Problem (QAP)* bertujuan meminimasi fasilitas/sumber dalam penugasannya.

2.4 Algoritma Semut

Any Colony System (ACS) awalnya dikembangkan oleh Marco Dorigo pada tahun 1991. Algoritma *Ant Colony System* (ACS) adalah algoritma yang didasarkan pada cara kerja semut untuk menentukan jarak terpendek dari sarang menuju sumber makanan. Semut dapat menemukan jarak terpendek dengan memanfaatkan jejak *pheromone* (air liur semut) yang dimanfaatkan sebagai komunikasi tidak langsung antar semut. Ketika semut berjalan, ia meninggalkan *pheromone* dalam jumlah tertentu pada jalur yang dilewatinya. Semut dapat mencium *pheromone* dan ketika memilih jalur mereka cenderung untuk memilih jalur dengan konsentrasi *pheromone* yang lebih besar adalah jarak terpendek.

Saat seekor semut yang terisolasi bergerak secara acak, semut ini akan mengikuti jejak yang telah ditinggalkan sebelumnya yang dapat dideteksi dan mempunyai tingkat probabilitas yang tinggi untuk diikuti dan melanjutkan jejak sebelumnya dengan *pheromone* baru. Tingkah laku kolektif yang muncul disebut dengan tingkah laku *Autocalystic*, dimana semut yang lain dapat mengikuti jejak yang ada dan jejak yang semakin jelas akan memudahkan bagi semut yang lain untuk

mengikutinya. Proses ini secara khusus terjadi melalui kumpulan umpan balik yang positif, dimana kemungkinan semut untuk memilih pola meningkat seiring dengan jumlah semut yang sebelumnya mengikuti pola yang sama.



Gambar 2.5 Jalur Solusi Semut (Ant Colony System)

Untuk menyelesaikan permasalahan algoritma *Ant Colony System*, berikut ini adalah rumus-rumus dasar yang dipakai dalam algoritma (rumus-rumus mengacu pada masalah TSP).

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum_{j\varepsilon} [\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}, \textit{jika } j \ \varepsilon \ (\Omega) \\ 0 \qquad \text{lainnya} \end{cases} \tag{2.1}$$

Keterangan :

$p_{ij}^{k}(t)$ **:** Probabilitas semut ke-*k* untuk pergi dari node atau kota *i* ke node *j* pada saat *t*.

$\tau_{ij}(t)$ **:** Intensitas *pheromone* pada jalur node *i* dan *j* pada saat *t*.

η_{ij} **:** Visibilitas/desirability untuk pergi dari node *i* ke node *j*.

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{2.2}$$

$\sum_{jE}$ **:** Jumlah *pheromone* yang menghubungkan node *i* ke node *j*.

α **:** Parameter yang menyatakan kepekaan terhadap jejak $(\alpha \rangle 0)$

β **:** Parameter yang menyatakan kepekaan terhadap *desirability* $(\beta \rangle 0)$.

d_{ij} **:** Jarak dari node *i* ke node *j*.

Ω **:** Himpunan node yang diperbolehkan dilewati dan boleh dilewati.

Setelah setiap node pada masing-masing kolom memiliki nilai probabilitas, maka akan dihitung probabilitas komulatif dengan persamaan dibawah ini

$$Q_{ij} = q_{ij} + Q_{i(j-1)} \quad (2.3)$$

Setelah didapatkan jalur terpendek, maka jalur tersebut akan di *update*. Jalur tersebut di *update* dengan tujuan untuk mengurangi jumlah *pheromone* pada semua sisi sebesar ρ dan menambah jumlah *pheromone* pada sisi yang memberikan jarak terpendek dari satu node ke node berikutnya sebesar $\Delta\tau_{ij}(t)$. Sedangkan sisi yang lain tidak mendapatkan penambahan *pheromone*. Berikut ini persamaan untuk meng*update* jumlah *pheromone*.

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \rho.\Delta\tau_{ij}(t) \quad (2.4)$$

Keterangan :

ρ : Parameter eveporasi/penguapan ($0< \rho <1$).

$\Delta\tau_{ij}(t)$: Penambahan jejek *pheromone*.

$\tau_{ij}(t+1)$: Intensitas *pheromone* pada saat *t + 1*.

Ada pula yang disebut *Tabu List* (π) yaitu sebuah matriks berukuran a (jumlah semut) x jumlah node. *Tabu List* ini berfungsi untuk sebagai memory semut, kemana saja semut tersebut sudah bergerak (node mana saja) sehingga jalur mana saja yang dilalui setiap semut dapat diketahui.

Algoritma umum ACS adalah sebagai berikut (mengacu pada masalah TSP):

Langkah 1 Inisialisasi

t = 0 ; Iterasi = *0*

untuk setiap jalur (*ij*) berikan *pheromone* awal (τ_0)

Langkah 2 Tempatkan a semut di node awal.

Langkah 3 Untuk semua semut.

Hitung probabilitas P_{ij} yang ada dengan persamaan (2.1)

Sampai semua node terpilih.

Langkah 4 Untuk setiap semut hitung panjang jalur yang telah ditempuh.

Langkah 5 Untuk setiap jalur (ij) terbaik lakukan perbaikan *pheromone* dengan menggunakan persamaan (2.3)

Langkah 6 Jika jumlah Iterasi Belum terpenuhi.
Maka kosongkan semua *TabuList*.
Kembali ke langkah 2.

2.4 Algoritma Hungarian

Algoritma Hungarian dikembangkan oleh Harold W Khun, pada tahun 1955. ini dibuktikan dengan dibuatnya paper yang ditulisnya sendiri yang berjudul "The Hungarian Method for The Assignment Problem". Algoritma yang Harold buat merupakan pengembangan atau hasil inspirasi dari dua orang ahli matematika yang berasal dari Hungaria, D Konig dan E Egervary. Oleh karena itulah maka Harold memberi nama Algoritma Hungarian.

Metode atau algoritma Hungarian adalah salah satu metode yang berfungsi untuk menyelesaikan masalah penugasan. Dalam algoritma Hungarian terdapat beberapa pengertian yang akan digunakan dalam penyelesaianmasalah *Quadratic Assignment Problem (QAP)*, yaitu :

1. *Opportunity-cost* (biaya kesempatan) / *Reduced-Cost Matrix*
   Pemilihan nilai terkecil pada setiap baris dari matriks biaya mula-mula untuk mengurangi seluruh elemen (bilangan) pada masing-masing baris.
2. *Reduced-cost matrix*
   Sama seperti *Opportunity-cost*, tetapi operasi pengurangan dilakukan pada kolom yang belum memiliki nilai nol.
3. *Total-Opportunity-Cost* nol
   Adanya kemungkinan setiap mesin sudah mendapatkan pekerjaan. Apabila ada yang belum mendapatkan pekerjaan maka matriks harus direvisi.