Penerapan *Hungarian Method* untuk Menyelesaikan *Personnel Assignment Problem*

Dian Perdhana Putra - NIM : 13507096
Program Studi Teknik Informatika
Insitut Teknologi Bandung
Jalan Ganesha 10 Bandung,
email: if17096@students.jika.itb.ac.id

Abstract *– Pada sebuah perusahaan, di mana terdapat n orang pekerja untuk n pekerjaan. Persoalan dimana setiap orang mendapatkan pekerjaan, satu orang untuk tiap pekerjaan, di mana pekerjaan tersebut merupakan kualifikasi dari pekerja yang melakukannya disebut "personnel assignment problem". Persoalan "personnel assignment problem" digunakan untuk menyelesaikan berbagai macam persoalan optimasi, misalnya meminimalisir biaya pekerja dalam sebuah proyek, mencari posisi seorang pemain yang tepat dalam sebuah tim sepak bola, mencari kemangkusan maksimal dari beberapa orang pekerja, dsb.*

Untuk menyelesaikan "personnel assignment problem" ini digunakan algoritma yang disebut "Hungarian method". Hungarian method dapat direpresentasikan dengan dua macam cara, yaitu dengan graf bipartit dan matriks. Representasi dengan graf bipartit biasanya digunakan untuk proses maksimasi, sebaliknya representasi dengan matriks digunakan untuk proses minimasi.

Kata Kunci*: personnel assignment problem, Hungarian method, optimasi, graf bipartit, matriks.*

1. PENDAHULUAN

Pada 1955, Harold Kuhn, seorang matematikawan Amerika mempublikasikan *Hungarian Method*. *Hungarian Method* adalah sebuah algoritma kombinasional untuk optimasi yang dapat digunakan untuk menemukan solusi optimal dari permasalahan *personnel assignment problem*. Algoritma ini diberi nama *Hungarian Method* untuk menghormati dua orang matematikawan asal Hungaria, yaitu Dénes Kőnig dan Jenő Egerváry.

Di tahun 1957, James Raymond Munkres seorang Professor Emeritus of mathematics dari MIT merevisi algoritma Kuhn dan oleh karena itu algoritma ini sering disebut *Kuhn Munkres algoritma*. Kompleksitas waktu dari algoritma ini adalah . Namun, Edmons dan Krap menemukan modifikasi untuk merubah kompleksitas waktu algoritma ini menjadi .

Ada dua macam interpretasi dari algoritma ini, yaitu dengan matriks dan graf bipartit. Matriks adalah susunan skalar elemen-elemen dalam bentuk baris dan kolom. Di dalam matriks **A** yang berukuran m baris dan n kolom (m x n), adalah elemen matriks pada baris ke- dan kolom ke- .

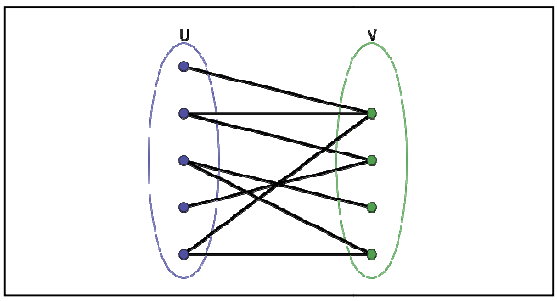
$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

Gambar 1: Matriks berukuran n x n

Graf merupakan pasangan himpunan dimana :

- = himpunan tidak kosong dari simpul-simpul (*simpul)* =
- = himpunan dari sisi-sisi (*sisis)* yang menghubungkan sepasang simpul =

Graf bipartit adalah graf dimana simpulnya dapat dibagi menjadi dua himpunan disjoint dan . Pada graf bipartit , setiap sisi pada menghubungkan sebuah simpul pada dengan sebuah simpul pada . Setiap pasang simpul pada , demikian pula pada , tidak bertetangga satu sama lain. Apabila setiap simpul pada bertetangga dengan setiap simpul pada , maka graf disebut graf bipartit lengkap.



Gambar 2: Graf bipartit

2. INTERPRETASI MATRIKS HUNGARIAN METHOD

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \dots & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \dots & c_{nn} \end{pmatrix}$$

Gambar 3: Matriks **C** berukuran n x n

Permasalahan *personnel assignment problem* banyak ditemui dalam kehidupan sehari-hari. Misalkan, dalam matriks di atas, c_{ij} = upah yang harus diberikan kepada pekerja ke-i untuk pekerjaan ke-j. Sebuah perusahaan ingin memperkerjakan n orang pekerja untuk menyelesaikan n buah pekerjaan dengan jumlah upah yang minimal. Untuk itu, perusahaan tersebut harus memilih tepat 1 elemen dari tiap baris dan tiap kolom, di mana jumlah keseluruhan dari elemen-elemen yang dipilih seminimal mungkin. Apabila matriks yang terbentuk bukanlah matriks bujursangkar, yaitu jumlah pekerja ≠ jumlah pekerjaan, maka harus dibentuk sebuah matriks perluasan dengan cara menambah baris atau kolom bernilai 0.

2.1. Algoritma Hungarian Method untuk matriks

1. Cari elemen tiap baris dalam matriks dengan nilai minimum. Kurangkan semua elemen dalam baris tersebut dengan elemen baris bernilai minimum.
2. Cari elemen tiap kolom dalam matriks dengan nilai minimum. Kurangkan semua elemen dalam baris tersebut dengan elemen kolom bernilai minimum.
3. Buat garis yang melingkupi semua elemen bernilai nol.
4. Lakukan pengecekan :
   a. Jika jumlah garis = n, maka pilih kombinasi dari elemen-elemen matriks, dimana jumlah kombinasi tersebut = 0.
   b. Jika jumlah garis < n, lakukan langkah ke-5.
5. Cari elemen dengan nilai minimum yang tidak terlingkupi oleh garis. Kurangkan elemen dengan nilai minimum dengan elemen-elemen yang tidak terlingkupi garis lainnya. Tambahkan elemen dengan nilai minimum dengan elemen-elemen yang terlingkupi garis horisontal dan garis vertikal. Apabila hasil pengurangan memberikan hasil negatif, maka elemen dengan nilai minimum tidak perlu dikurangkan. Kembali ke langkah ke-3.
6. Catatan : Apabila persoalan yang dihadapi adalah persoalan memaksimalkan, kalikan matriks **C** dengan skalar -1.

Contoh persoalan :

Sebuah perusahaan konstruksi bangunan memiliki empat buah buldoser. Keempat buah buldoser itu memiliki jarak yang berbeda-beda terhadap lokasi pembangunan seperti terlihat dalam tabel berikut :

Tabel 1. Tabel Jarak Buldoser-Lokasi Pembangunan

		Jarak ke Lokasi Pembangunan (km)			
		1	2	3	4
Buldoser	1	90	75	75	80
	2	35	85	55	65
	3	125	95	90	105
	4	45	110	95	115

Tempatkan masing-masing buldoser pada sebuah lokasi pembangunan dimana jumlah keseluruhan jarak ke lokasi pembangunan dibuat seminimal mungkin.

Solusi dari permasalahan tersebut adalah :

1. Bentuk matriks berukuran 4 x 4, dan kemudian cari elemen dengan nilai minimum pada tiap baris.

$$\begin{pmatrix} 90 & \mathbf{75} & 75 & 80 \\ \mathbf{35} & 85 & 55 & 65 \\ 125 & 95 & \mathbf{90} & 105 \\ \mathbf{45} & 110 & 95 & 115 \end{pmatrix}$$

   Kurangkan elemen dengan nilai minimum pada tiap baris dengan semua elemen pada baris tersebut.

$$\begin{pmatrix} 15 & 0 & 0 & 5 \\ 0 & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{pmatrix}$$

2. Cari elemen dengan nilai minimum pada tiap kolom.

$$\begin{pmatrix} 15 & \mathbf{0} & \mathbf{0} & \mathbf{5} \\ \mathbf{0} & 50 & 20 & 30 \\ 35 & 5 & 0 & 15 \\ 0 & 65 & 50 & 70 \end{pmatrix}$$

   Kurangkan elemen dengan nilai minimum pada tiap kolom dengan semua elemen pada kolom tersebut.

$$\begin{pmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{pmatrix}$$

3. Buat garis yang melingkupi semua elemen bernilai nol.

$$\begin{pmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & 20 & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{pmatrix}$$

4. Lakukan pengecekan :
   (jumlah garis =3) < (n = 4)
   maka, lanjutkan ke langkah ke-5.

5. Cari elemen dengan nilai minimum yang tidak terlingkupi oleh garis. Dalam matriks ini, elemen tersebut adalah (2,3) dengan nilai 20.

$$\begin{pmatrix} 15 & 0 & 0 & 0 \\ 0 & 50 & \mathbf{20} & 25 \\ 35 & 5 & 0 & 10 \\ 0 & 65 & 50 & 65 \end{pmatrix}$$

   Kurangkan elemen dengan nilai minimum yang tidak terlingkupi garis tersebut dengan elemen-elemen yang tidak terlingkupi garis lainnya. Tambahkan elemen dengan nilai minimum dengan elemen-elemen yang terlingkupi garis vertikal **dan** horisontal (yaitu elemen (1,1) dan (3,1)).

$$\begin{pmatrix} 35 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 \\ 55 & 5 & 0 & 10 \\ 0 & 45 & 30 & 45 \end{pmatrix}$$

   Kembali ke langkah ke-3.

$$\begin{pmatrix} 35 & 0 & 0 & 0 \\ 0 & 30 & 0 & 5 \\ 55 & 5 & 0 & 10 \\ 0 & 45 & 30 & 45 \end{pmatrix}$$

   Lakukan pengecekan :
   (jumlah garis = 3) < (n =4)

   Lakukan kembali langkah ke-5 :
   Cari elemen dengan nilai minimum yang tidak terlingkupi oleh garis. Dalam matriks ini, elemen tersebut adalah (2,4) dengan nilai 5.

$$\begin{pmatrix} 35 & 0 & 0 & 0 \\ 0 & 30 & 0 & \mathbf{5} \\ 55 & 5 & 0 & 10 \\ 0 & 45 & 30 & 45 \end{pmatrix}$$

   Kurangkan elemen dengan nilai minimum dengan elemen-elemen yang tidak terlingkupi garis lainnya. Tambahkan elemen dengan nilai minimum dengan elemen-elemen yang terlingkupi garis vertikal dan garis horisontal (yaitu elemen (1,1) dan (1,3)).

$$\begin{pmatrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{pmatrix}$$

Kembali ke langkah ke-3.

$$\begin{pmatrix} 40 & 0 & 5 & 0 \\ 0 & 25 & 0 & 0 \\ 55 & 0 & 0 & 5 \\ 0 & 40 & 30 & 40 \end{pmatrix}$$

Lakukan pengecekan :
(jumlah garis = 4) = (n =4)

$$\begin{pmatrix} 40 & \mathbf{0} & 5 & \mathbf{0} \\ \mathbf{0} & 25 & \mathbf{0} & \mathbf{0} \\ 55 & \mathbf{0} & \mathbf{0} & 5 \\ \mathbf{0} & 40 & 30 & 40 \end{pmatrix}$$

Kita lihat posisi dari elemen-elemen yang bernilai 0. Posisi dari elemen bernilai 0 adalah kombinasi yang mungkin. Dalam kasus ini, kita mempunyai dua kemungkinan kombinasi :

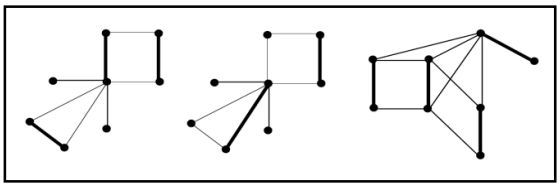
1. Buldoser 1 : Lokasi pembangunan 4 (80 km)
   Buldoser 2 : Lokasi pembangunan 3 (55 km)
   Buldoser 3 : Lokasi pembangunan 2 (95 km)
   Buldoser 4 : Lokasi pembangunan 1 (45 km)
   Jumlah jarak = (80 + 55 + 95 + 45) km
   = 275 km

2. Buldoser 1 : Lokasi pembangunan 2 (75 km)
   Buldoser 2 : Lokasi pembangunan 4 (65 km)
   Buldoser 3 : Lokasi pembangunan 3 (90 km)
   Buldoser 4 : Lokasi pembangunan 1 (45 km)
   Jumlah jarak = (75 + 65 + 90 +45)km
   = 275 km

3. INTERPRETASI GRAF BIPARTIT HUNGARIAN METHOD

3.1. *Matching*

Matching M pada graf *G* adalah himpunan dari sisi-sisi pada *G*, dimana tidak satupun di antaranya adalah loop,yaitu tidak ada dua sisi pada *M* yang mempunyai titik akhir yang sama. *Matching M saturates* simpul *u* pada *G*, maka *u* disebut *M-saturated*, jika *u* adalah titik akhir dari sisi pada *M*; jika tidak , *u* disebut *M-unsaturated*. Simpul *u* dan *v* dikatakan *matched* dalam *M* jika $uv \in M$.
Matching pada *G* yang *saturates* setiap simpulnya dikatakan *perfect matching*. *Matching M* pada *G* dengan jumlah terbesar sisi dibandingkan dengan *matching* lainnya pada *G* disebut *maksimum matching*.



Gambar 4: Maksimum *matching* dan bukan *perfect* , maximal *matching* yang bukan maksimum, dan *perfect matching* pada graf.

Misalkan M adalah *matching* dan P adalah lintasan pada graf G. lintasan P disebut *M-alternating* jika sisi-sisi pada P terbentang dalam M dan berada pada $E(G) - M$. lintasan P dengan sedikitnya satu sisi disebut *M-augmenting* jika lintasan ini *M-alternating* dan kedua titik akhir dari P merupakan *M-unsaturated*.

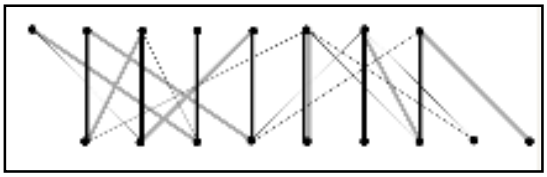
Teorema Berge *Matching M pada graf G akan maksimum jika dan hanya jika G tidak mengandung lintasan M-augmenting.*

Bukti : Asumsikan M merupakan *maximum matching* pada G. Misalkan, G mengandung *lintasan M-augmenting* $P = v_0v_1\ldots v_k$. Perhatikan bahwa k jumlah sisi pada P) harus berjumlah ganjil karena setiap sisi lain melewati M, dan v_0v_1 dan $v_{k-1}v_k$ keduanya tidak melewati M.

Didefinisikan himpunan sisi-sisi $M' \subseteq (G)$ dimana

$$M' = (M - \{v_1v_2;\ v_3v_4,\ \ldots\ ,\ v_{k-2}v_{k-1}\}) \cup \{v_0v_1,\ v_2v_3, \ldots,\ v_{k-1}vk\} \ldots\ldots\ldots\ldots (1)$$

Maka, M' merupakan *matching* pada G dengan nilai $|M'| = |M| + 1$, yang berkontradiksi dengan maksimalitas dari M. Oleh karena itu, G tidak mungkin mempunyai lintasan *M-augmenting*.



Gambar 5: Dua *matching M* dan *M**pada graf G: komponen terhubung dari ketidaksamaan mereka, $G[M \oplus M^*]$ adalah lintasan dan sirkuit dengan sisi yang *alternating* antara

Untuk himpunan $S \subseteq V(G)$ dari simpul pada graf G , didefinisikan *neighbourhood* dari S sebagai
$NG(S) = \{x \in V(G) : x \sim G\ u$ untuk beberapa $u \in S\}$.

Teorema Hall *Misalkan G adalah graf bipartit dengan bipartition* $(X;\ Y)$. *Maka G mengandung sebuah*
matching yang memenuhisetiap simpul di X jika dan hanya jika
$|N(S)| \geq |S|$ *untuk setiap* $S \subseteq X$.

Bukti : Asumsikan G mengandung *matching* M yang saturates tiap simpul di X. Misalkan S adalah subset dari X. Karena tiap simpul in S matched ke simpul di $N(S)$ dalam M, dan karena dua simpul berbeda di S matched ke dua simpul berbeda di $N(S)$, kita mempunyai $|N(S)| \geq |S|$.
Asumsikan G memenuhi (*), dan misalkan G tidak mempunyai *matching* yang saturates tiap simpul dari X. Misalkan, M^* menjadi maksimum *matching* dari$G$. Maka, akan terdapat simpul u di X yang M^*-unsaturated. Didefinisikan himpunan simpul in G:

$Z = \{v \in V(G)$: akan terdapat lintasan M^*- alternating dari u ke $v\}$, dimana
$S = Z \cap X$
$T = Z \cap Y$

Karena M^* is a *maksimum matching* dan u is M^*-unsaturated, oleh Teorema Berge , u adalah satu-satunya simpul *M*-unsaturated* dalam Z (sebaliknya, kita akan memiliki sebuah lintasan *M*-augmenting*, berkontradiksi dengan maksimalitas dari M^*). Oleh karena itu, tiap simpul in $S - \{u\}$ matched dalam M^* ke sebuah simpul unik dalam T dan sebaliknya, dan jika $|T| = |S| - 1$.
Oleh definisi dari S dan T, terlihat jelas $N(S) = T$. namun kemudian $|N(S)| = |T| = |S| - 1 < |S|$,berkontradiksai dengan asumsi (*).

Corollary *Jika G adalah k-regular graf bipartit dengan* $k > 0$, *maka G mempunyai perfect matching.*

Bukti : Misalkan G menjadi *k-regular* graf bipartit with bipartition $(X;\ Y)$. Karena G k-regular, kita memiliki $|E| = |X|$ dan, lain kata,$|E| = k|Y|$. Karena $k > 0$, kita dapat menyimpulkan bahwa $|X| = |Y|$.
Kita dapat membuktikan kondisi cukup (*) dari Teorema Hall dipenuhi. Misalkan S adalah subset of X. Lebih jauh lagi, misalkan E_S dan $E_{N(S)}$ adalah set dari sisi-sisi yang bertetangga dengan simpul pada S dan $N(S)$. Dari definisi $N(S)$, kita mendapatkan $E_S \subseteq E_{N(S)}$.
Oleh karenanya,

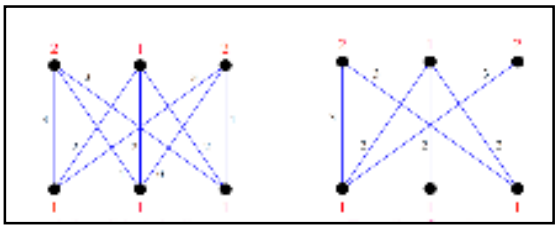
$$k|N(S)| = |E_{N(S)|} \geq |E_S| = k|S| \ldots\ldots\ldots\ldots\ldots (2)$$

ini menunjukkan bahwa $|N(S)| \geq |S|$ untuk setiap $S \subseteq X$. Dari Teorema Hall, didapatkan bahwa G memiliki *matching* M yang memenuhi tiap simpul di X. Tapi, karena $|X| = |Y|$, *matching* tersebut pastilah *perfect matching* .
Corollary di atas sering disebut sebagai *the marriage theorem*, karena colloraly ini dapat diutarakan dengan statemen :

Jika setiap gadis di sebuah desa mengenal tepat sejumlah k pemuda dan setiap pemuda mengenal tepat sejumlah k gadis, maka tiap gadis dapat menikahi seorang pemuda yang dia tahu dan tiap pemuda dapat menikahi seorang gadis yang dia tahu.

Misalkan *G* adalah graf bipartit dengan bipartition (*X; Y*). Dalam algoritma, kita dapat membentuk sebuah *matching M* yang menyatukan tiap simpul dalam set *X*,atau sebalinya menentukan sebuah *matching* tidak ada.
Algoritma bermula dari *matching* sembarang *M*. Jika *M* saturates tiap simpul in *X*, then *M* adalah *matching* yang diharapkan. Jika tidak, kita pilih sebuah *M-unsaturated* simpul *u* in *X* dan secara sistematik mencari lintasan *M-augmenting* yang bermula di *u*. Jika lintasan tersebut ada , maka lintasan ini akan digunakan untuk membangun *matching* yang lebih besar dari *M*.

Feasible Labelings dan Equality Grafs



Gambar 6: Feasible Labelings dan Equality Grafs

Feasible Labelings adalah
$\ell(x)+\ell(y) \geq w(x, y),\ \forall x \in X, y \in Y$……(3)

Equality Graf (*with respect to* ℓ) adalah

$$G = (V,E_\ell) \text{ dimana}$$
$$E_\ell = \{(x, y) : \ell(x)+\ell(y) = w(x, y)\}\text{.....(4)}$$

Teorema Kuhn-Munkres : Jika ℓ adalah *feasible labelling* dan M adalah *perfect matching* pada E_ℓ maka M merupakan *maximum weight matching*.

Unutk menemukan initial *feasible labeling* :

$$\forall y \in Y, \ell(y) = 0$$
$$\forall x \in X, \ell(x) = max_{y \in Y}\{w(x, y)\}\text{.....(5)}$$

Sehingga
$\forall x \in X, y \in Y, w(x) \leq \ell(x)+\ell(y)$.....(6)

Improving Labellings
Misalkan ℓ adalah *feasible labelling*.
Didefinisikan *neighbor* dari u ∈ V dan set S ⊆ V adalah
$N\ell(u) = \{v : (u, v) \in E_\ell, \}$, $N_\ell(S) = \cup_{u \in S} N_\ell(u)$……(7)

Lemma: Misalkan S ⊆ X and T = $N_\ell(S) \neq Y$, maka
$\alpha_\ell = min_{x \in S, y \notin T}\{\ell(x) + \ell(y) - w(x, y)\}$……(8)
dan

$$\ell'(v) = \begin{cases} \ell(v) - \alpha\ell \text{ if } v \in S \\ \ell(v) + \alpha\ell \text{ if } v \in T \\ \ell(v) \text{ otherwise} \end{cases} \text{.....(9)}$$

Maka ℓ' adalah *feasible labeling* dan
(i) jika (x, y) ∈ E_ℓ untuk x ∈ S, y ∈ T then
(x, y) ∈ $E_{\ell'}$.
(ii) jika (x, y) ∈ Eℓ untuk x ∉ S, y ∉ T maka
(x, y) ∈Eℓ' .
(iii) terdapat beberapa sisi (x, y) ∈ $E_{\ell'}$ untuk x ∈ S, y ∉T.

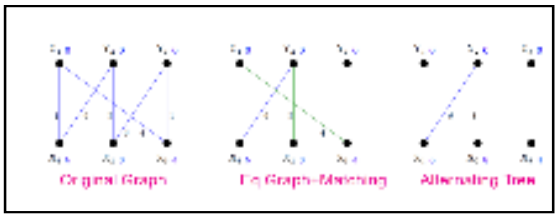
3.2 Algoritma Hungarian Method pada graf

1. Generate initial labelling ℓ and *matching* M in E_ℓ.
2. If M *perfect* , stop. Otherwise pick free vertex u ∈ X. Set S = {u}, T = ∅.
3. If Nℓ(S) = T, update labels (forcing Nℓ(S) 6= T):

$$\alpha_\ell = \min_{x \in S, y \notin T}\{\ell(x) + \ell(y) - w(x, y)\}$$
$$\ell'(v) = \begin{cases} \ell(v) - \alpha\ell \text{ if } v \in S \\ \ell(v) + \alpha\ell \text{ if } v \in T \\ \ell(v) \text{ otherwise} \end{cases}$$

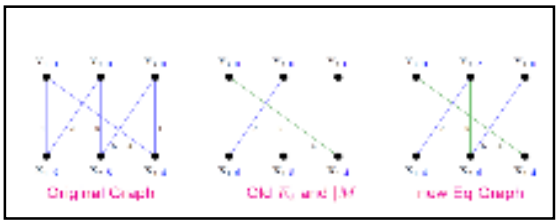
4. If $N_\ell(S) \neq T$, pick y ∈ $N_\ell(S)$ − T. If y free, u − y is augmenting path. Augment M and go to 2. If y matched, say to z, extend alternating tree:S = S ∪ {z}, T = T ∪ {y}. Go to 3.

Contoh :



Gambar 7: Graf awal, equivalent graf dan *matching*, alternating tree.

1. Inisialisasi graf, trivial labelling, dan bentuk equality graf.
2. Inisialisasi *matching* : (x3, y1), (x2, y2).
3. S = {x_1}, T = ∅.
4. Karena $N_\ell(S) \neq T$, lakukan langkah 4 dalam algoritma.
   Pilih y2 ∈ $N_\ell(S)$ − T.
5. y2 matched jadi bentuk tree dengan menambah (y2, x2), sehingga., S = {x1, x2}, T = {y2}.
6. Pada poin ini $N_\ell(S)$ = T, jadi lanjutkan ke langkah 3 dalam algoritma.

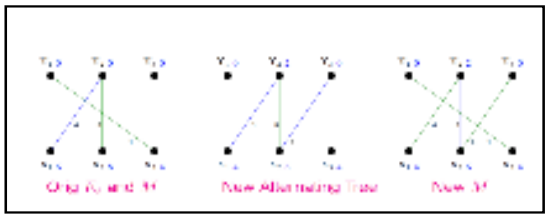


Gambar 8: Graf awal, E_l lama dan |M|, equivalent graf baru.

7. $S = \{x_1, x_2\}$, $T = \{y_2\}$ dan $N_\ell(S) = T$
8. Hitung α_ℓ

$$\alpha_\ell = \min_{x \in S, y \notin T} \begin{cases} 5+0-1, & (x_1, y_1) \\ 5+0 \quad 0, & (x_1, y_3) \\ 8 \mid 0-0, & (x_2, y_1) \\ 3+0-6, & (x_2, y_3) \end{cases}$$
$$= 2$$

9. Kurangi label dari S dengan 2, tambahkan label dari T dengan 2.
10. Sekarang, $N\ell(S) = \{y2, y3\}\ 6= \{y2\} = T$.



Gambar 9: E_l lama dan M, alternating tree baru, M baru.

11. $S = \{x1, x2\}$, $N\ell(S) = \{y2, y3\}$, $T = \{y2\}$
12. Pilih $y_3 \in N\ell(S) - T$ dan tambahkan ke T.
13. y3 tidak matched di M jadi, kita telah menemukan alternating path x1, y2, x2, y3 dengan dua titik akhir bebas. Oleh karena itu, kita dapat memperluas M untuk mendapatkan *matching* yang lebih besar pada equality graf yang baru. *Matching* ini *perfect* , jadi merupakan *matching* optimal.
14. Perhatikan bahwa *matching* (x1, y2), (x2, y3), (x3, y1)mempunyai jumlah 6 + 6 + 4 = 16 yaitu jumlah dari label di final feasible labelling.

Perhatikan bahwa *Hungarian Method* tidak secara langsung membentuk maksimum *matching* dalam graf bipartit *G*. Yaitu, jika algoritma ini berhenti dengan *N*(*S*) = *T*, kemudian *matching M* yang dihasilkan tidak secara langsung maksimum karena mungkin ada *Lintasan M-augmenting* dalam bagian lain dari graf. Pada beberapa kasus, bangun pohon *M-alternating* dari tiap simpul *M-unsaturated* dalam *X* hingga *matching* saturating *X* terbentuk. Pada kasus lain, *matching* yang terbentuk merupakan maksimum (namun tidak saturate *X*).

Misalkan *G* adalah graf bipartit dengan bipartition (*X; Y*). Dalam algoritma, kita dapat membentuk sebuah *matching M* yang saturates tiap simpul dalam set *X*,atau sebaliknya menentukan sebuah *matching* tidak ada.
Algoritma bermula dari *matching* sembarang *M*. Jika *M* saturates tiap simpul in *X*, maka *M* adalah *matching* yang diharapkan. Jika tidak, kita pilih sebuah *M-unsaturated* simpul *u* in *X* dan secara sistematik mencari l*intasan M-augmenting* yang bermula di *u*. Jika lintasan tersebut ada , maka lintasan ini akan digunakan untuk membangun *matching* yang lebih besar dari *M.*

Sebaliknya, himpunan *Z* dari titik-titik akhir dari semua *lintasan M-alternating* bermula di *u* ditemukan, dan dalam bukti dari Teorema Hall, himpunan $S = Z \cap X$ memenuhi $|N(S)| < |S|$ sehingga *matching* yang menyatukan semua simpul in *X* ada.

Pencarian dari lintasan *M-augmenting* bermula dari *u* dengan membentuk pohon *M-alternating H* yang berakar di *u*. Yaitu , *H* adalah pohon pada graf *G* dengan kriteria sebagai berikut :

1. $u \in V(H)$ dan
2. untuk setiap $v \in V(H)$, $(u; v)$-lintasan unik pada *H* adalah lintasan *M-alternating*.

Misalkan $S = V(H) \cap X$ dan $T = V(H) \cap Y$. Pohon *M-alternating H* tumbuh pada dua kondisi yaitu :
(i) semua simpul in $S - \{u\}$ adalah *M-saturated* dan matched dalam *M* ke simpul pada *T*, atau
(ii) *T* mengandung sebuah simpul *M-unsaturated* simpul *y* berbeda dari *u*.

Pada saat kasus (ii) dipenuhi, kita akan memiliki simpul *M*-augmenting $(u; y)$, yang digunakan untuk memperbesar *matching M*. Pada kasus (i), kita memiliki $T \subseteq N(S)$. Jika $T = N(S)$, maka $|Sj| = |T| + 1 > |N(S)|$, dan *G* tidak mempunyai *matching* saturating *X*; algoritma berhenti.Namun jika $T \neq N(S)$, maka terdapat simpul $y \in N(S) - T$ yang bertetangga ke sebuah simpul *x'* idi *S*. Jika *y* is *M*-unsaturated, maka sisi *x'y* akan ditambahkan ke pohon *H*, menghasilkan Kasus (ii). Sebaliknya, jika *y M*-saturated, maka terdapat sisi $xy \in M$ with $x \notin S$ (karena semua simpul pada $S - \{u\}$ sudah dalam *M* ke simpul pada *T*). Kita sekarang dapat menambahkan *x'y* dan *yx* ke the pohon *H*, menghasilkan kasus (ii).

4. KESIMPULAN

Hungarian method dapat digunakan untuk menyelesaikan *personnel assignment problem*. *Hungarian method* dapat direpresentasikan dengan dua macam cara, yaitu dengan graf bipartit dan matriks.
Dalam representasi matriks, matriks haruslah berbentuk matriks bujursangkar berukuran n x n. Jika tidak berbentuk bujursangkar, maka harus dibentuk sebuah matriks perluasan dengan cara menambah baris atau kolom bernilai 0.

Dalam representasi graf, *matching M* pada graf *G* adalah himpunan dari sisi-sisi pada *G*, dimana tidak satupun di antaranya adalah loop,yaitu tidak ada dua sisi pada *M* yang mempunyai titik akhir yang sama. *Matching* M pada graf G akan maksimum jika dan hanya jika G tidak mengandung lintasan *M-augmenting*. Jika ℓ adalah *feasible labelling* dan M adalah *perfect matching* pada E_ℓ maka M merupakan *maximum weight matching*.

5. UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir dan Ibu Harlili untuk bimbingannya dalam mata kuliah Struktur Diskrit sehingga makalah ini dapat diselesaikan.

DAFTAR REFERENSI

[1] Munir, Rinaldi. (2004). Bahan Kuliah IF2091 Struktur Diskrit. Departemen Teknik Informatika,
Institut Teknologi Bandung.

[2] Xu, Junming. (2003). *Theory and Application of Graphs*. Kluwer Academic Publishers, London.

[3] Dingzhu Du(1999). *Handbook of Combinatorial Optimization: Supplement Volume A*. Springer.

[4] http://www.ee.oulu.fi/~mpa/matreng/eem1_2-1.htm,
diakses tanggal 2 Januari 2009 pukul 18.00

[5] http://www.math.fau.edu/locke/Courses/Rec-Math/Kuhn.htm, diakses tanggal 31 Desember 2008, pukul 17.00

[6] http://www.ee.oulu.fi/~mpa/matreng/ematr12 .htm, diakses tanggal 31 Desember 2008 pukul 17.30

[7] http://en.wikipedia.org/wiki/Hungarian_algorithm , diakses tanggal 31 Desember 2008 pukul 18.00