PENYELESAIAN *MATCHING* BOBOT MAKSIMUM MASALAH PENUGASAN DENGAN METODE HUNGARIAN

skripsi

disajikan sebagai salah satu syarat

untuk memperoleh gelar Sarjana Sains

Program Studi Matematika

oleh

Hanifah Putri Wijaya
4150405518

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS NEGERI SEMARANG

2009

PERNYATAAN KEASLIAN TULISAN

Dengan ini saya menyatakan bahwa skripsi ini tidak terdapat karya yang pernah diajukan untuk memperoleh gelar sarjana di suatu Perguruan Tinggi, dan sepanjang pengetahuan saya tidak terdapat karya yang diterbitkan oleh orang lain, kecuali yang secara tertulis dirujuk dalam skripsi ini dan disebutkan dalam daftar pustaka.

Semarang,

Hanifah Putri Wijaya

NIM. 4150405518

PENGESAHAN

Skripsi ini telah dipertahankan di hadapan sidang Panitia Ujian Skripsi Fakultas Ilmu Pengetahuan Alam Universitas Negeri Semarang pada tanggal 25 Agustus 2009.

Panitia:

Ketua,	Sekretaris,
Dr. Kasmadi Imam S., M.S	Drs. Edy Soedjoko, M.Pd
NIP. 130781011	NIP. 131693657
	Penguji,
	Mulyono, S.Si, M.Si
	NIP. 132158717
Penguji/Pembimbing I	Penguji/Pembimbing II
Drs. Amin Suyitno, M.Pd	Isnaini Rosyida, S.Si, M.Si
NIP. 130604211	NIP. 132205927

MOTTO DAN PERSEMBAHAN

MOTTO

- *"Sesungguhnya sesudah kesulitan itu ada kemudahan. Maka, apabila kamu telah menyelesaikan satu urusan, kerjakanlah dengan sungguh-sungguh urusan yang lain"*
  (QS. Al-Insyirah: 6-7).
- Orang berakal tidak akan bosan untuk meraih manfaat berpikir, tidak putus asa dalam menghadapi keadaan, dan tidak akan pernah berhenti dari berpikir dan berusaha.
- Orang yang selalu bersyukur akan selalu merasakan nikmat Tuhan dan akan selalu merasa bahagia.

PERSEMBAHAN

- ♥ Special thank's for all My Family, Ayah dan Bunda tercinta. Kakakqu tercantik Novida dan adekku termanis Yaya. Keluarga Raharjo, Papah Susilo dan Mamah Mugiyati, dek Bayu, dek Widya, dek Yogi. Keluarga Batang dan Sragen. Eyangku tersayang.
- ♥ My Risky. Penyemangatku.
- ♥ Dosen idolaku, Bapak Amin Suyitno dan Ibu Isnaini Rosyida yang telah membimbing saya selama pembuatan skripsi.
- ♥ Teman seperjuanganku, Marom, Fitri.
- ♥ Sahabatku, Nindi, Fatma. Teman baekQu, Dwi Ari, Dewi, Intan, Iva, Sinta, Deq yan, Zulfa, Endra, Salman, Tini, Hani, Dian, Aqib, Seftian, Eko, Fera, Yayat, Titis.
- ♥ Teman kosku, Siska, Sekar, Nisa, Mimi, Garin, Tia, Kiki, Novi, Mudah, Reta.
- ♥ Almamaterku.

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan limpahan rahmat dan hidayah-Nya, sehingga penulis memperoleh kekuatan untuk menyelesaikan skripsi ini. Dalam kesempatan ini penulis menghaturkan terima kasih yang tak terhingga kepada:

1. Prof. Dr. H. Sudijono Sastroatmodjo, M.Si., Rektor Universitas Negeri Semarang yang telah memberikan fasilitas-fasilitas kepada penulis.
2. Dr. Kasmadi Imam S., M.S, Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Semarang.
3. Drs. Edy Soedjoko, M.Pd, Ketua Jurusan Matematika Universitas Negeri Semarang
4. Drs. Amin Suyitno, M.Pd, Dosen Pembimbing I yang penuh keikhlasan mengarahkan dan membimbing penulis dalam menyusun skripsi ini dari awal hingga akhir.
5. Isnaini Rosyida, S.Si, M.Si, Dosen Pembimbing II yang penuh keikhlasan mengarahkan dan membimbing penulis dalam menyusun skripsi ini dari awal hingga akhir.
6. Bapak dan Ibu dosen Jurusan Matematika yang telah memberikan bekal ilmu dan pengetahuan selama kuliah.
7. Bapak Priyono dan Ibu Dewi, kedua orang tuaku yang telah dengan sabar dan ikhlas mencurahkan waktu untuk mendidik, memberi kasih sayang, menasihati, dan membimbing penulis.
8. Teman-teman Matematika Angkatan 2005 yang telah memberikan dukungannya hingga terselesaikannya skripsi ini.

Semoga Allah SWT senantiasa memberikan balasan atas bantuan dan amal baiknya. Penulis berharap semoga skripsi ini dapat bermanfaat bagi para pembaca.

Semarang, Agustus 2009

Penulis

ABSTRAK

Wijaya, Hanifah Putri. 2009. *Penyelesaian Matching Bobot Maksimun Masalah Penugasan dengan Metode Hungarian*. Skripsi. Jurusan Matematika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Semarang. Pembimbing I: Drs. Amin Suyitno, M.Pd dan Pembimbing II: Isnaini Rosyida, S.Si, M.Si.

Kata kunci: *matching* bobot maksimum penugasan

Salah satu topik yang menarik dalam graf adalah masalah penugasan pekerjaan yang dapat dimodelkan dalam graf bipartit G (S, T, E) yang himpunan partisinya cocok dengan pekerja dan tempat/ kedudukan. Masalah penugasan optimal yang merupakan *matching* bobot maksimal dalam graf bipartit ini dapat menggunakan algoritma Hungarian. Metode yang digunakan dalam algoritma Hungarian dapat memecahkan masalah sangat sederhana dan mudah dipahami. Solusi yang diperoleh bila kita menggunakan algoritma Hungarian akan selalu optimum. Berdasarkan penjelasan di atas, penulis ini mengkaji bagaimana menyelesaikan *matching* bobot maksimum masalah penugasan dengan metode Hungarian.

Penelitian ini merupakan penelitian studi pustaka yang dilakukan dalam dua tahap, yaitu (1) mempelajari dan mengkaji *matching* bobot maksimum masalah penugasan dalam graf bipartit dengan menggunakan referensi yang ada serta bagaimana membuktikan teorema yang mendukung keberadaannya (2) menggunakan metode Hungarian untuk menyelesaikan *matching* bobot maksimum masalah penugasan dalam graf bipartit.

Untuk menyelesaikan *matching* bobot maksimum masalah penugasan dalam graf bipartit dapat digunakan algoritma Hungarian. Sebelum membahas algoritma Hungarian terlebih dahulu dibahas teorema: suatu *matching* M dalam graf G adalah *matching* maksimum jika dan hanya jika tidak ada *path* perluasan yang memuat M dalam G. Langkah-langkah yang digunakan untuk menyelesaikan *matching* bobot maksimum masalah penugasan sebagai berikut. Memodelkan permasalahan tersebut dengan program linier, menyelesaikan bentuk dual dari program linier pada langkah sebelumnya, menyelesaikan model dual tersebut dengan algoritma Hungarian. Solusi yang diperoleh bila kita menggunakan algoritma hungarian akan selalu optimum.

Berdasarkan hasil penelitian tersebut, penulis menyarankan kepada peneliti lain untuk mengkaji penyelesaian *matching* bobot maksimum masalah penugasan dalam graf bipartitel lengkap dengan algorit lain

DAFTAR ISI

BAB I

PENDAHULUAN

1.1 Latar Belakang

Salah satu topik yang menarik dalam graf adalah masalah penugasan pekerjaan. Misalnya, diberikan suatu pekerja (sumber) dengan satu atau beberapa pekerja untuk satu atau beberapa tempat/ kedudukan (tujuan). Situasi ini dapat dimodelkan dengan graf bipartit G (S, T, E) yang himpunan partisinya cocok dengan pekerja dan tempat/ kedudukan. Sebuah graf bipartit G(S, T, E) adalah suatu graf G(X, E) yang himpunan *vertex* X dapat dibagi menjadi dua himpunan S dan T sedemikian sehingga tidak ada sisi di dalam graf yang menghubungkan dua *vertex* pada himpunan bagian yang sama. Masalah penugasan yang memerlukan jumlah maksimal dari posisi yang dipenuhi misalnya, perusahaan penyewa menginginkan untuk mendapatkan keuntungan maksimal sebagai hasil dari penyewaan. Sebagai contoh, pengalaman pekerja merupakan faktor penting yang dipertimbangkan selama proses penyewaan. Perusahaan beruntung dengan mempekerjakan sedikit orang tetapi banyak pengalaman daripada jumlahnya besar dengan pengalaman sedikit. Untuk menggambarkan situasi tersebut digunakan graf bipartit (S, T, E) dengan S= banyaknya orang (pekerja) dan T= banyaknya pengalaman. Masalah penugasan ini disebut masalah penugasan optimal yang merupakan *matching* bobot maksimal dalam graf bipartit. Salah satu metode yang dapat digunakan untuk memecahkan masalah *matching* bobot maksimum dengan menggunakan algoritma Hungarian. Algoritma Hungarian adalah salah

satu algoritma yang digunakan untuk menyelesaikan persoalan masalah *assignment*. Algoritma ini sendiri diciptakan oleh Harold Kuhn dan James Munkres, yang terinspirasi oleh hasil penemuan dua orang matematikawan asal Hungaria, yaitu Denes Konig dan Jeno Evergary. Metode yang digunakan dalam algoritma Hungarian dapat memecahkan masalah sangat sederhana dan mudah dipahami. Solusi yang diperoleh bila kita menggunakan algoritma Hungarian akan selalu optimum. Berdasarkan alasan di atas, penulis tertarik untuk mengkaji penyelesaian *matching* bobot maksimum masalah penugasan dengan metode Hungarian.

1.2 Perumusan Masalah

Permasalahannya adalah sebagai berikut. Bagaimana menyelesaikan *matching* bobot maksimum masalah penugasan dengan metode Hungarian?

1.3 Pembatasan Masalah

Pada skripsi ini masalah yang akan dibahas hanya pada graf bipartit yang tidak berarah.

1.4 Tujuan dan Manfaat Kegiatan

Tujuan kegiatan ini adalah mempelajari penyelesaian *matching* bobot maksimum masalah penugasan dalam graf bipartit dengan algoritma Hungarian. Sedangkan manfaat kegiatan adalah sebagai berikut.

1.4.1 Bagi penulis

Untuk mengembangkan ilmu pengetahuan, terutama dalam menyelesaikan masalah penugasan.

1.4.2 Bagi pembaca

Untuk menambah ilmu pengetahuan terutama dalam hal penyelesaian *matching* bobot maksimum masalah penugasan dalam graf bipartit dengan algoritma baru.

1.5 Sistematika Penulisan

BAB I Merupakan bab pendahuluan yang berisi tentang latar belakang, perumusan masalah, pembatasan masalah, tujuan dan manfaat kegiatan.

BAB II Menguraikan materi penunjang yang menjadi dasar teori disusunnya skripsi ini.

BAB III Merupakan bab metode penelitian berisi langkah-langkah yang dilakukan peneliti.

BAB IV Menguraikan pembahasan tentang Penyelesaian *matching* bobot maksimum masalah penugasan dengan metode Hungarian.

BAB V Berisi Kesimpulan dan Saran dari pembahasan tentang Penyelesaian *matching* bobot maksimum masalah penugasan dengan metode Hungarian.

BAB II

LANDASAN TEORI

2.1 Definisi Graf

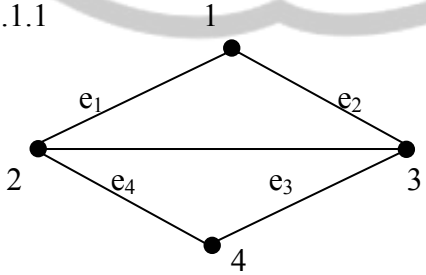
G disebut graf jika G terdiri dari dua himpunan yaitu himpunan hingga tak kosong V(G) yang elemen-elemennya disebut *vertex* dan himpunan (mungkin kosong) E(G) yang elemen-elemennya disebut sisi, sedemikian hingga setiap elemen e dalam E(G) adalah sebuah pasangan tak berurutan dari *vertex-vertex* di V(G). V(G) disebut himpunan *vertex* dari G dan E(G) disebut himpunan sisi dari G.

Definisi 2.1.1

Misal u dan v adalah *vertex-vertex* G dan sisi e = (u, v) (sering ditulis e = uv = vu) adalah sisi dari G. Dikatakan, sisi e menghubungkan *vertex-vertex* u dan v; *vertex* u dan *vertex* v bertetangga (*adjacent*) di G; u dan v adalah *vertex-vertex* akhir dari sisi e; sisi e bersisian (*incident*) dengan *vertex* u atau v.

Order adalah banyaknya *vertex-vertex* dalam sebuah graf G.

Contoh 2.1.1



Gambar 2.1.1

Pada gambar 2.1.1, *vertex* 1 bertetangga dengan *vertex* 2 dan 3, tetapi tidak *adjacent* dengan *vertex* 4.

Sedangkan sisi (2, 3) bersisian dengan *vertex* 2 dan *vertex* 3, sisi (2, 4) *incident* dengan *vertex* 2 dan *vertex* 4, tetapi sisi (1, 2) tidak bersisian dengan *vertex* 4.

Definisi 2.1.2

Loop adalah sisi yang menghubungkan suatu *vertex* dengan dirinya sendiri.

Definisi 2.1.3

Jika terdapat lebih dari satu sisi yang menghubungkan dua *vertex*, maka sisi-sisi tersebut dinamakan sisi rangkap.

Definisi 2.1.4

Dua buah sisi disebut bertetangga jika kedua sisi tersebut mempunyai salah satu *vertex* yang sama.

Definisi 2.1.5

Banyaknya sisi yang bersisian dengan suatu *vertex* v_i (loop dihitung dua kali) disebut derajat (*degree*) dari *vertex* tersebut, dinotasikan $d(v_i)$. Derajat minimum dari graf G dinotasikan dengan $\delta(G)$ dan derajat maksimumnya dinotasikan dengan $\Delta(G)$.

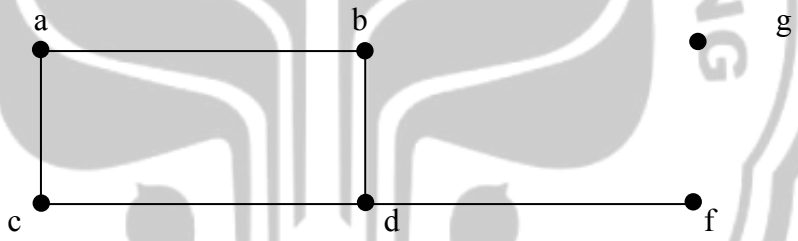
Contoh 2.1.5

Pada gambar 2.1.1, d (v_1) = d (v_4) = 2 dan d (v_2) = d (v_3) = 3. Jadi, δ(G) = 2 dan Δ(G) = 3.

Definisi 2.1.6

Suatu *vertex* berderajat 0 disebut sebagai suatu *vertex* terisolasi (*vertex* terasing), sedangkan *vertex* berderajat 1 merupakan *end-vertex* (*vertex* akhir).

Contoh 2.1.6



Gambar 2.1.5

g merupakan *vertex* terasing karena mempunyai derajat 0 dan f merupakan *vertex* akhir karena mempunyai derajat 1.

Definisi 2.1.7

Suatu *vertex* dikatakan genap atau ganjil berdasarkan derajat *vertex* tersebut genap atau ganjil.

Contoh 2.1.7

Pada gambar 2.1.5 maka: - a,b,c dan g merupakan *vertex* genap.

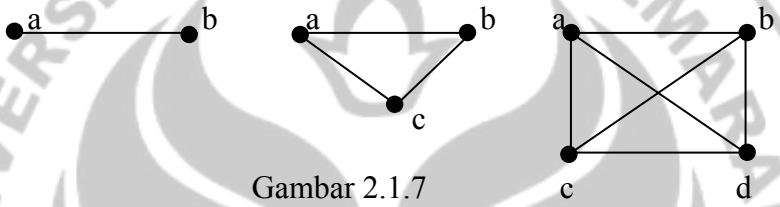
- d dan f merupakan *vertex* ganjil.

Definisi 2.1.8

Suatu graf dikatakan reguler jika semua *vertex* dari graf tersebut mempunyai derajat yang sama.

Contoh 2.1.8

Graf reguler derajat 1, 2 dan 3:



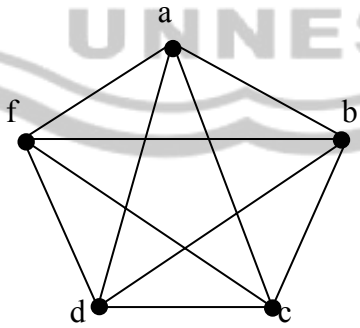
Gambar 2.1.7

Definisi 2.1.9

Sebuah graf G dikatakan r-*reguler* atau *reguler* berderajat r, jika setiap *vertex* pada G mempunyai derajat r.

Contoh 2.1.9

G:



Gambar 2.1.8

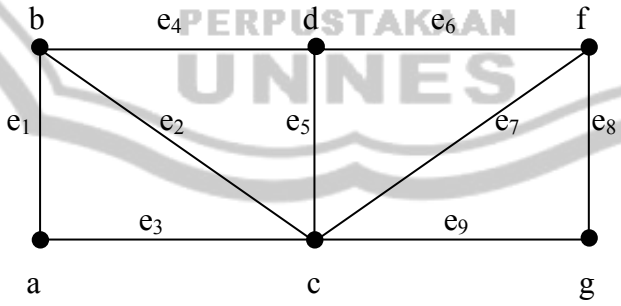
Graf G merupakan graf 4-*reguler*.

2.2 *Walk, Trail* dan *Path*

Definisi 2.2.1

Misalkan G graf. Sebuah jalan (*walk*) di G adalah sebuah barisan berhingga (tak kosong) $W = v_0\ e_1\ v_1\ e_2\ v_2 ... e_k\ v_k$ yang suku-sukunya bergantian *vertex* dan sisi, sedemikian sehingga v_{i-1} dan v_i adalah *vertex* akhir sisi e_i, untuk $1< i < k$. Kita katakan W adalah sebuah jalan dari v_0 ke v_k. *Vertex* v_0 dan *vertex* v_k berturut-turut disebut *vertex* awal dan *vertex* akhir W. Sedangkan *vertex-vertex* $v_1, v_2,...,v_{k-1}$ disebut *vertex-vertex* internal dari W; dan k disebut panjang dari W. Perhatikan bahwa panjang dari jalan W adalah banyaknya sisi dalam W. Jika semua sisi $e_1, e_2, e_3,...,e_k$ dalam jalan W berbeda, maka W disebut sebuah jejak (*trail*).

Contoh 2.2.1



Gambar 2.2.1

a e_1 b e_4 d e_5 c e_2 b e_4 d e_6 f e_7 c adalah perjalanan antara a dan c yang memuat sisi e_4 dua kali, *vertex* b, c dan d dua kali.

Definisi 2.2.2

Jika semua *vertex* dan sisi dalam jalan W juga berbeda, maka W disebut sebuah lintasan (*path*). *A nontrivial path* adalah *path* yang mempunyai *order* sekurang-kurangnya 2.

Contoh 2.2.2

Pada gambar 2.2.1 perjalanan a e_1 b e_2 c e_7 f e_8 g tidak ada *vertex* dan sisi yang diulang karena itu merupakan *path*.

Definisi 2.2.3

Closed walk adalah sebuah jalan yang berawal dan berakhir pada *vertex* yang sama.

Open walk adalah sebuah jalan yang berawal dan berakhir pada *vertex* yang tidak sama.

Closed trail adalah sebuah jejak yang berawal dan berakhir pada *vertex* yang sama.

Definisi 2.2.4

Sebuah sikel (*cycle*) adalah sebuah jejak tertutup (*closed trail*) yang *vertex* awal dan semua *vertex* internalnya berbeda. Suatu *cycle* genap (*even cycle*) adalah *cycle* yang memuat bilangan genap dari sisi tersebut

dan suatu *cycle* ganjil (*odd cycle*) adalah *cycle* yang memuat bilangan ganjil dari sisi tersebut.

Banyaknya sisi dalam suatu *cycle* disebut panjang dari *cycle* tersebut.

cycle dengan panjang k disebut k- *cycle*.

Definisi 2.2.5

Panjang lintasan adalah banyaknya sisi dalam lintasan tersebut.

Contoh 2.2.5

Lintasan a e_1 b e_2 c e_7 f e_8 g pada gambar 2.1.1 memiliki panjang 4.

Definisi 2.2.6

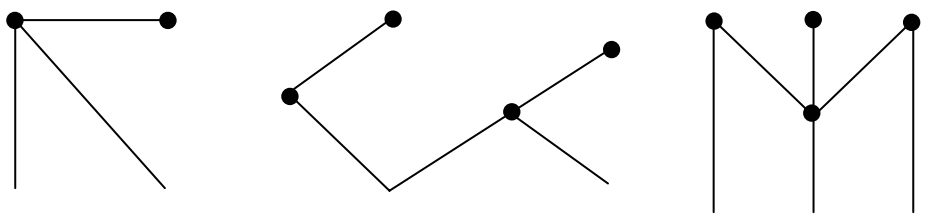
Graf tak-berarah G disebut graf terhubung (*connected graph*) jika untuk setiap pasang *vertex* v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j (yang juga harus berarti ada lintasan dari v_j ke v_i).

Definisi 2.2.7

Pohon (*tree*) adalah graf terhubung yang tidak memuat sikel (cycle).

Contoh 2.2.7

Tiga buah graf pada gambar 2.2.7 adalah pohon.



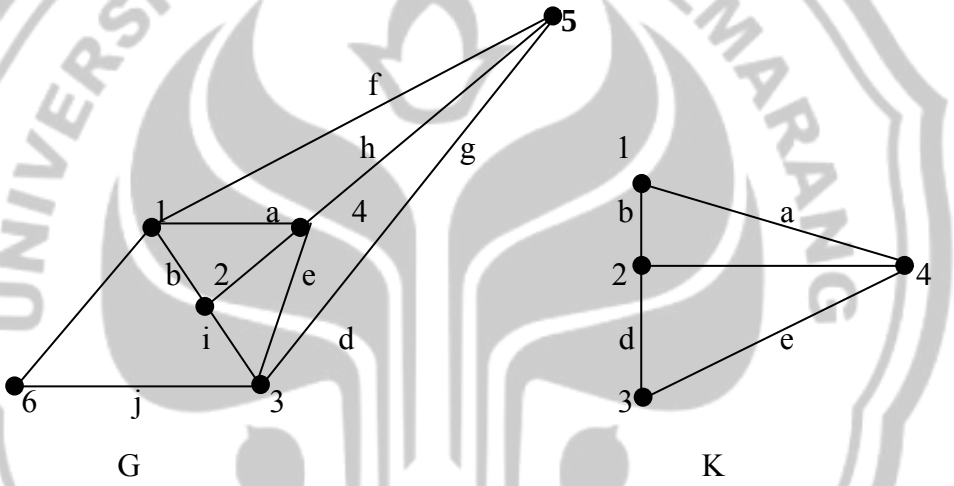
Gambar 2.2.7

2.3 Definisi Subgraf

Definisi 2.3.1

Sebuah graf K disebut graf bagian (subgraf) dari graf G, dinotasikan $K \subseteq G$, jika $V(K) \subseteq V(G)$ dan $E(K) \subseteq E(G)$.

Contoh 2.3.1



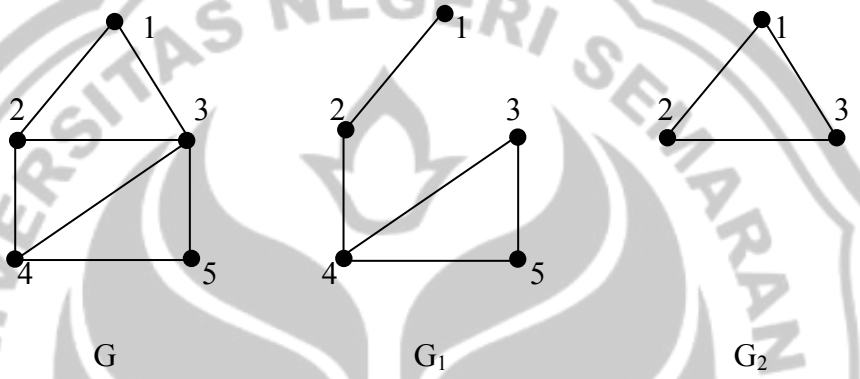
Gambar 2.3.1

Definisi 2.3.2

Subgraf $G_1 = (V_1, E_1)$ dari $G = (V, E)$ dikatakan *spanning* subgraf jika $V_1 = V$.

Contoh 2.3.2

Pada gambar 3.2, G_1 adalah *spanning* subgraf dari G, tetapi G_2 bukan *spanning subgraph* dari G karena G_2 tidak mengandung semua *vertex* G.



Gambar 2.3.2

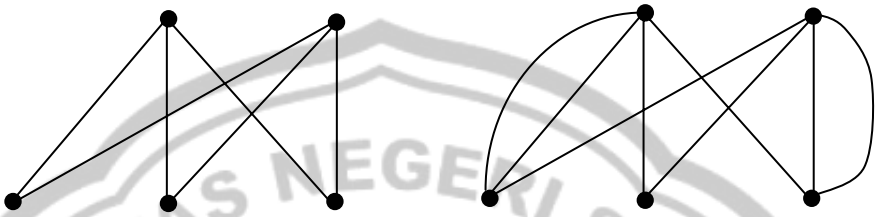
2.4 Graf khusus

Definisi 2.4.1

Sebuah graf G disebut graf bipartit jika V(G) dapat dipartisi menjadi dua himpunan bagian S dan T sedemikian sehingga setiap sisi dari G menghubungkan sebuah *vertex* di S dan sebuah *vertex* di T. Kita notasikan (S, T) bipartit dari G.

Graf bipartit G dapat dinotasikan juga dengan G (S, T, E).

Contoh 2.4.1

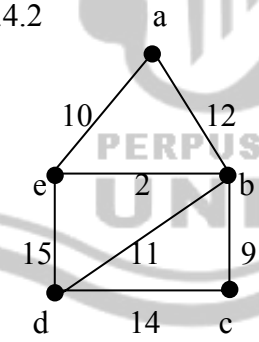


Gambar 2.4.1

Definisi 2.4.2

Graf berbobot adalah sebuah graf G dimana setiap sisi e adalah suatu bilangan real *non negative*, w(e) disebut bobot dari e. Bobot dari graf G dinotasikan w(g) yaitu jumlah bobot dari semua sisi-sisi.

Contoh 2.4.2



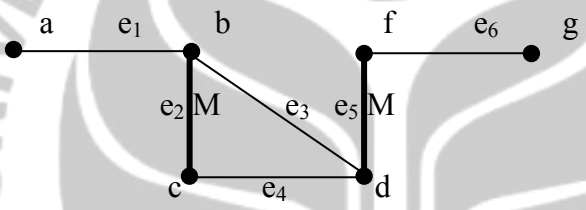
Gambar 2.4.2. Graf bebobot.

2.5 Matching

Definisi 2.5.1

Suatu *matching* dalam graf adalah suatu himpunan sisi dengan aturan tidak ada *vertex* yang *incident* dengan lebih dari satu sisi dalam himpunan.

Contoh 2.5.1



Gambar 2.5.1

Dalam gambar 2.5.1 $M_1 = \{e_2, e_5\}$ dan $M_2 = \{e_1, e_4, e_6\}$ merupakan *matching*.

Definisi 2.5.2

Suatu *single vertex* adalah *vertex* yang tidak berada dalam sebuah *matching*.

Contoh 2.5.2

Single vertex dalam *matching* M_1 dari gambar 5.1 adalah a dan g.

Definisi 2.5.3

Suatu *path* berayun (*alternating path*) adalah *path* yang sisiya berada dalam sebuah *matching* maupun yang tidak dalam sebuah *matching*. Jika *path* berayun dimulai dan diakhiri dengan *single vertex* disebut dengan *path* perluasan (*augmenting path*).

Contoh 2.5.3

Pada gambar 5.1 yang merupakan *path* berayun adalah abcdf, abcdfg, bcdf, bcdfg. Sedang *path* perluasan pada gambar 5.1 adalah abcdfg.

Definisi 2.5.4

Sebuah *vertex* dikatakan *vertex matched* jika *vertex* tersebut merupakan *vertex* dalam sebuah *matching*, sedang sebuah sisi dikatakan sisi *matched* jika sisi tersebut memuat *matched* dan suatu sisi disebut sisi *unmatched* jika sisi tersebut tidak memuat *matched*.

Contoh 2.5.4

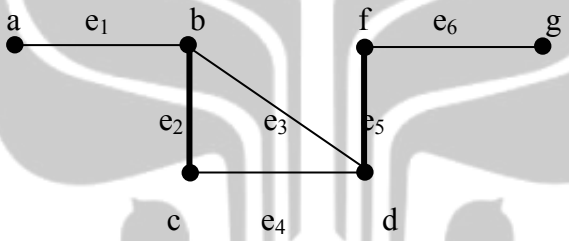
Pada gambar 5.1, b, f, c, d merupakan *vertex matched* dari *matching* M_1, e_2 dan e_5 merupakan sisi *matched* dari *matching* M_1, sedang e_1, e_3, e_4, e_6 disebut sisi *unmatched matching* M_1.

2.6 *Matching* maksimum dan *matching* bobot maksimum

Definisi 2.6.1

Kardinalitas suatu *matching* adalah banyaknya sisi dalam *matching* tersebut, dinotasikan $|\ |$. Sebuah *matching* graf G dengan kardinalitas maksimum disebut *matching* maksimum.

Contoh 2.6.1



Gambar 2.6.1

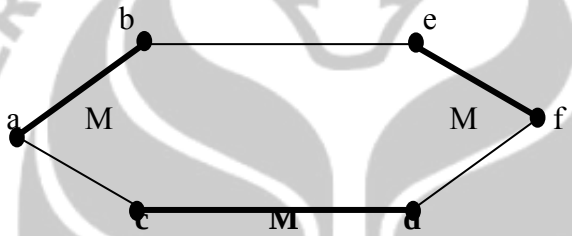
Dalam graf G di atas himpunan M_1 = {bc, fd} merupakan *matching* sedang M_2 = {ab, cd, fg} merupakan *matching* maksimum, karena kardinalitas M_2 maksimum, yaitu:

$|M_1| = 2$ dan $|M_2| = 3$.

Definisi 2.6.2

Jika G merupakan graf dengan *order* P yang mempunyai *matching* dengan kardinalitas p/2, maka *matching* tersebut disebut *perfect matching*.

Contoh 2.6.2



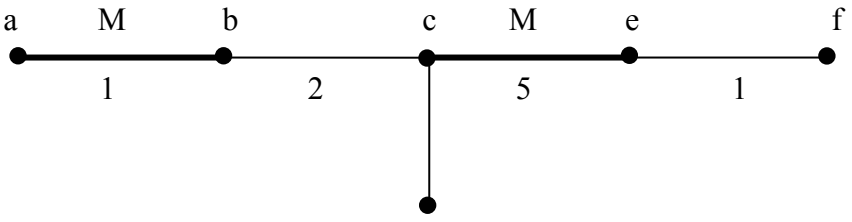
Gambar 6.2

Graf di atas mempunyai order 6, sehingga *perfect matching* mempunyai kardinalitas 6/2 = 3, yaitu {ab, cd, ef}, {ac, df, eb}.

Definisi 2.6.3

Matching bobot maksimum dalam graf berbobot merupakan suatu *matching* yang jumlah dari bobot sisinya maksimum.

Contoh 2.6.3



1

d

Gambar 2.6.3

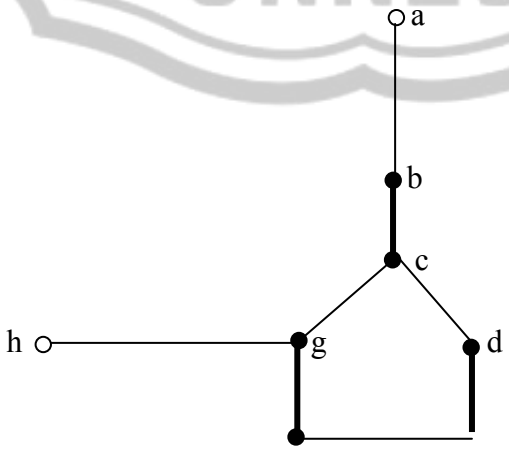
Pada gambar 2.6.3 menunjukkan bahwa *matching* M* = {ab, ce} merupakan *matching* bobot maksimum, sedangkan M' = {bc, cd, ef} bukan merupakan *matching* bobot maksimum, karena δ(M*) =6 dan

δ (M') =3.

2.6.4 Perluasan *Matching* sekitar *Path* Perluasan

Definisi. Misal M *matching* dalam graf G, dan andaikan bahwa P sebuah *path* perluasan yang memuat *matching* M. Misal M' menunjukkan himpunan sisi dari P yang termasuk sisi-sisi pada M, dan misal M'' = E(P) – M'. Himpunan $M_1 = (M - M') \cup M''$ dan M_1 adalah sebuah *matching* dari G yang mempunyai kardinalitas $|M|+1$. Dikatakan pula M_1 dihasilkan dengan memperluas M sepanjang P.

Contoh 2.6.4:



Gambar 2.6.4.a

Dari graf G di atas didapat hasil sebagai berikut:

M = {bc, de, fg}.

P = {a b c d e f g h}.

M' = {bc, de, fg}.

M'' = E(P) – M'

= {ab, bc, cd, de, ef, fg, gh}-{bc, de, fg}

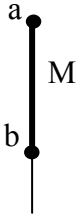
= {ab, cd, ef, gh}.

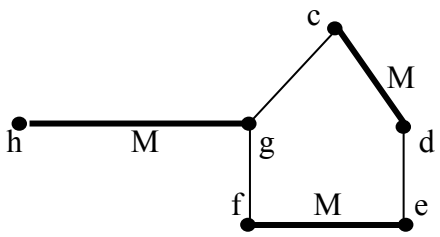
M_1 = (M – M') ∪ M''

= {(bc, de ,fg) – (bc, de, fg) – (ab, cd, ef, fg)}

= {ab, cd, ef, gh}.

M_1 merupakan *matching* yang dihasilkan dengan perluasan *matching* sekitar *path* perluasan P yang digambarkan pada gambar berikut:





Gambar 2.6.4.b

Gambar 2.6.4.b merupakan perluasan *matching* sekitar *path* perluasan. Dalam gambar 2.6.4.a menunjukkan sisi G dan *matching* M. *Vertex matched* ditebalkan, sedangkan *single vertex* tidak ditebalkan. Gambar 2.6.4.b menunjukkan graf G dan *matching* M_1, yang dihasilkan dengan memperluas M disekitar *path* perluasan P: a b c d e f g h.

Teorema 2.6.4.1

Misal M_1 dan M_2 *matching* di graf G. Misal E himpunan sisi di G yang termasuk sisi-sisi di M_1 atau di M_2, tetapi tidak untuk keduanya. Bahwa $E = (M_1 - M_2) \cup (M_2 - M_1)$. Misal H *spanning subgraph* dari G dengan himpunan sisi E (H = E). Maka masing-masing komponen H adalah salah satu dari tipe:

(a) Sebuah vertex terasing

(b) Sebuah cycle genap yang sisinya berurutan dalam M_1 dan M_2

(c) Sebuah *nontrivial path* yang sisinya berurutan dalam M_1 dan M_2 sedemikian sehingga masing-masing *vertex* akhir dalam *path* tersebut merupakan *vertex* di M_1 atau M_2, tetapi tidak untuk keduanya.

Bukti:

Pertama tinjau bahwa $\Delta(H) \leq 2$ karena setiap *vertex* H *incident* dengan paling banyak satu sisi M_1 dan paling banyak satu sisi M_2. Akibatnya, setiap komponen H merupakan sebuah *path* (kemungkinan *trivial*) atau sebuah *cycle.* Jika sebuah komponen H adalah *trivial path*, maka komponen itu adalah sebuah *vertex* terasing. Berikutnya perhatikan komponen H yang memuat sisi. Karena tidak ada dua sisi dalam *matching* yang *adjacent*, maka sisi-sisi dari setiap *cycle* dan *path* H merupakan sisi-sisi di M_1 dan M_2. Oleh karena itu, setiap *cycle* H adalah genap.

Andaikan e = u v adalah sebuah sisi H dan u adalah *vertex* akhir dari suatu *path* P yang merupakan sebuah komponen H. Bukti akan menjadi lengkap jika dapat ditunjukkan bahwa u adalah sebuah *vertex* tunggal yang terkait dengan satu *vertex* di M_1 dan M_2. Karena $e \in E(H)$ maka $e \in M_1 - M_2$ atau $e \in M_2 - M_1$. Misalakan $e \in M_1 - M_2$. Maka u adalah sebuah *vertex matched* yang terkait dengan M_1. Selanjutnya, akan dibuktikan u adalah *vertex* tunggal yang terkait juga dengan satu titik di M_2. Andaikan ada sebuah sisi f dalam M_2 (jadi , $f \neq e$) sedemikian sehingga f *incident* dengan u. Karena e dan f *adjacent*, $f \in M_1$. Jadi $f \in M_2 - M_1 \subseteq E(H)$. Hal ini tidak mungkin karena u adalah sebuah *vertex* akhir dari P. Kontradiksi, sehingga u adalah *vertex*

tunggal yang terkait dengan sebuah titik di M_2. Dengan demikian, jika e $\in M_2 - M_1$, maka u adalah tunggal yang memuat M_1.

Teorema 2.6.4.2

Suatu *matching* M dalam graf G adalah *matching* maksimum jika dan hanya jika tidak ada *path* perluasan yang memuat M dalam G.

Bukti:

($\Rightarrow$) Jika M *matching* maksimum dalam G maka tidak ada *path* perluasan yang memuat M dalam G.

Misal M *matching* maksimum dalam G. Andaikan G ada *path* perluasan P maka P mempunyai panjang ganjil. Misal M' menunjukkan himpunan sisi-sisi P yang memuat M dan misal M'' = E(P) – M'. Karena $|M''| = |M'| + 1$ maka himpunan $(M - M') \cup M''$ adalah sebuah *matching* yang kardinalitasnya lebih besar dari M. Kontradiksi, karena diketahui bahwa M adalah *matching* maksimum. Sehingga yang benar adalah jika M merupakan *matching* maksimum maka tidak ada *path* perluasan yang memuat M.

($\Leftarrow$) Jika tidak ada *path* perluasan yang memuat M dalam G maka M adalah *matching* maksimum.

Misal M_1 *matching* dari graf G dan tidak ada *path* perluasan yang memuat M_1 dalam G. Akan dibuktikan bahwa M_1 adalah *matching* maksimum. Misal M_2 *matching* maksimum dalam G. Dari bukti bagian pertama, tidak ada *path* perluasan yang memuat M_2. Misal H *spanning subgraph* dari G dengan $E(H) = (M_1 - M_2) \cup (M_2 - M_1)$. Akan dibuktikan bahwa setiap komponen H memuat sisi-sisi yang berjumlah genap dalam sisi, dimana setengah berada di M_1 dan setengah lagi berada di M_2.

Misal H_1 komponen dari H. Maka H_1 memenuhi salah satu tipe (a), (b) atau (c) pada Teorema 2.6.4.1. Jika H_1 merupakan tipe (a) maka diabaikan, karena tidak dapat digunakan untuk membuktikan pernyataan. Jika H_1 memenuhi tipe (b) atau H_1 terdiri dari *cycle* genap yang sisinya berurutan dalam M_1 dan M_2 maka jelas banyaknya sisi di H_1 genap dimana setengahnya berada di M_1 dan setengahnya berada di M_2. Jika H_1 memenuhi tipe (c) atau H_1 *path* berayun yang sisinya berurutan M_1 dan M_2 sedemikian sehingga masing-masing *vertex* akhir dalam *path* tersebut merupakan *vertex* di M_1 atau M_2, maka H_1 memiliki sisi berjumlah genap dimana setengahnya berada di M_1 dan setengahnya berada di M_2.

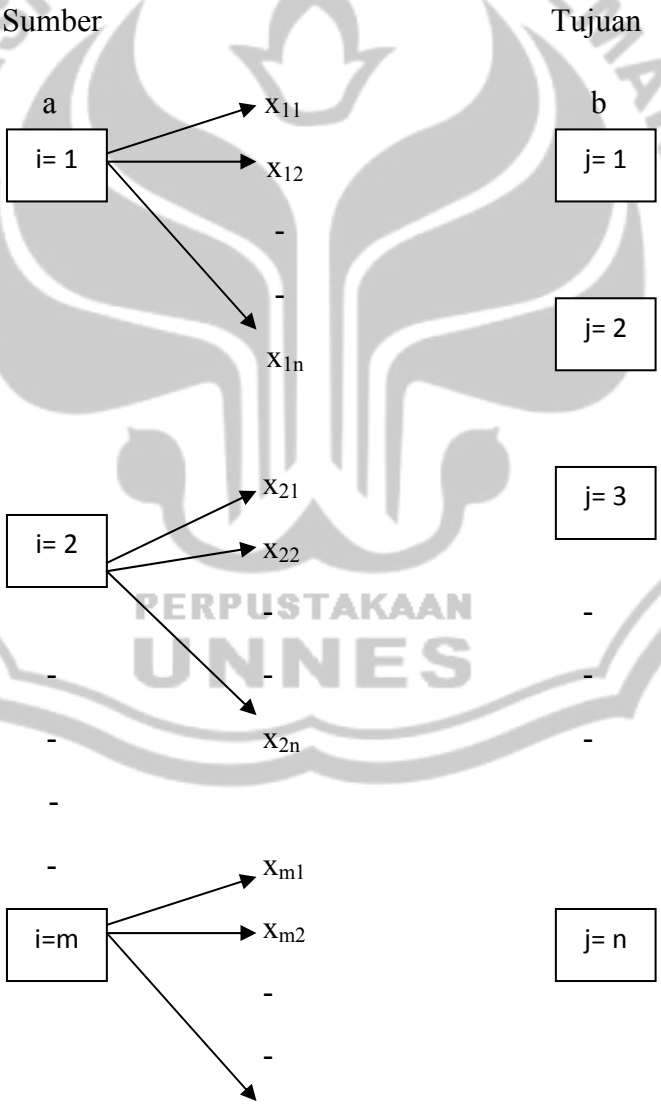
Jadi dapat dibuktikan bahwa $|M_1| = |M_2|$ atau M_1 merupakan *matching* maksimum.

2.7 Model Transportasi

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

Secara dramatik, model transportasi dapat digambarkan sebagai berikut:

Misalkan ada m buah sumber dan n buah tujuan.



x_{mn}

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

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

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

d) Ongkos pengiriman per *unit* dari sumber i ke tujuan j adalah c_{ij}.

Dengan demikian, maka formulasi program liniernya adalah sebagai berikut:

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

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

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

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

2.8 Model Penugasan

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

Biasanya yang dimaksud dengan sumber ialah pekerjaan (atau pekerja), sedangkan yang dimaksud dengan tujuan ialah mesin-mesin. Jadi, dalam hal ini, ada m pekerjaan yang ditugaskan pada n mesin, di

mana apabila pekerjaan i (i = 1, 2,..., m) ditugaskan kepada mesin j (j = 1, 2,..., n) akan muncul ongkos penugasan c_{ij}. Karena satu pekerjaan ditugaskan hanya pada satu mesin, maka *supply* yang dapat digunakan pada setiap sumber adalah 1 (atau $a_i = 1$, untuk seluruh i). Demikian 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). Jika ada suatu pekerjaan yang tidak dapat ditugaskan pada mesin tertentu, maka c_{ij} yang berkorespondensi dengannya merupakan ongkos yang sangat tinggi. Model penugasan ini merupakan penggambaran *matching* bobot maksimum masalah penugasan dalam graf bipartit.

Penggambaran umum persoalan penugasan tersebut adalah sebagai berikut:

		Mesin				
		1	2	. . .	n	
	1	c_{11}	c_{12}	. . .	c_{1n}	1
Pekerjaan	2	c_{21}	c_{22}	. . .	c_{2n}	1
	.	.	.		.	.
	.	.	.		.	.
	.	.	.		.	.
	m	c_{m1}	c_{m2}	. . .	c_{mn}	1

1 1 . . . 1

Secara matematis, model penugasan ini dapat dinyatakan sebagai berikut:

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

Dengan demikian, model persoalan penugasan ini adalah:

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

Berdasarkan pembatas:

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

$$\sum_{i=1}^{m} x_{ij} = 1, \; j = 1,2,\ldots,n$$

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

BAB III

METODE PENELITIAN

Langkah-langkah yang dilakukan dalam penelitian ini sebagai berikut.

1. Penemuan Masalah

Metode ini merupakan tahapan pertama dalam penelitian yaitu dengan pencarian ide atau gagasan materi dari bidang kajian yang dipilih dan dijadikan permasalahan untuk dikaji pada penelitian ini.

2. Perumusan Masalah

Perumusan masalah dalam skripsi ini adalah bagaimana menentukan *matching* bobot maksimum pada masalah penugasan dalam graf bipartit untuk graf tidak berarah?

3. Studi Pustaka

Studi pustaka merupakan penelaah sumber pustaka relevan yang digunakan untuk mengumpulkan data maupun informasi yang diperlukan dalam penelitian ini. Studi pustaka diawali dengan mengumpulkan sumber pustaka yaitu berupa buku-buku maupun referensi yang menjadi dasar dalam penelitian ini. Setelah sumber pustaka terkumpul dilanjutkan dengan penelaahan isi sumber pustaka tersebut. Pada akhirnya sumber pustaka ini dijadikan landasan untuk melakukan penelitian ini.

4. Analisis Pemecahan Masalah

Pada tahap ini dilakukan analisis dan pemecahan masalah yaitu dengan langkah-langkah sebagai berikut.

a. Mempelajari dan mengkaji *matching* bobot maksimum masalah penugasan dalam graf bipartit dengan menggunakan referensi yang ada serta bagaimana membuktikan teorema yang mendukung keberadaannya.
b. Membahas alur Metode Hungarian.
c. Menggunakan metode Hungarian untuk menyelesaikan *matching* bobot maksimum masalah penugasan dalam graf bipartit.

5. Penarikan Simpulan

Tahap ini merupakan tahap terakhir dari penelitian. Setelah menganalisis dan memecahkan masalah berdasarkan studi pustaka dan pembahasannya kemudian dibuat sebagai simpulan sebagai jawaban dari permasalahan yang telah dirumuskan sebelumnya.

BAB IV

PENYELESAIAN *MATCHING* BOBOT MAKSIMUM MASALAH PENUGASAN DENGAN METODE HUNGARIAN

Algoritma adalah suatu himpunan aturan atau instruksi yang didefinisikan secara jelas untuk mendapatkan keluaran tertentu dari masukan tertentu pula dalam jumlah berhingga langkah. Algoritma yang digunakan untuk menyelesaikan masalah penugasan dalam graf bipartit ini antara lain Algoritma Hungarian. Sebelum membahas algoritma Hungarian terlebih dahulu akan dibahas.

4.1 Pemodelan masalah penugasan *matching* bobot maksimum dalam graf bipartit

Masalah penugasan dapat dimodelkan dalam graf bipartit G(S, T, E). Penentuan *matching* bobot maksimum pada graf G dapat dimodelkan dengan program linier sebagai berikut.

Memaksimalkan: $\sum_i \sum_j c_{ij} x_{ij}$ (4.1.1)

Kendala: $\sum_j x_{ij} \leq 1$, untuk semua i $\in$ S (4.1.2)

$\sum_i x_{ij} \leq 1$, untuk semua j $\in$ T (4.1.3)

$x_{ij} \geq 0$, untuk semua i $\in$ S, j $\in$ T (4.1.4)

Keterangan:

c_{ij} = matriks masalah penugasan (bobot tiap sisi)

x_{ij} = variabel sisi dari *vertex* i ke *vertex* j

$\sum_j x_{ij} \leq 1$ artinya paling banyak hanya ada satu sisi x_{ij} $\forall$ i $\in$ S yang terpilih sebagai sisi *matching.*

$\sum_i x_{ij} \leq 1$ artinya paling banyak hanya ada satu sisi x_{ij} $\forall$ j $\in$ T yang terpilih sebagai sisi *matching.*

$x_{ij} \geq 0$ artinya banyaknya sisi minimal 0 $\forall$ i $\in$ S, j $\in$ T

Masalah ini sama seperti bentuk masalah transportasi khusus. Titik optimalnya selalu ada dalam bentuk *integer*. Kemudian x_{ij} = 1 jika dan hanya jika sisi dari iϵS sampai jϵT termasuk dalam *matching* optimal (maksimum). Batasan yang jelas adalah lebih dari satu sisi dalam solusi fisibel *incident* ke beberapa *vertex*.

Dalam mengerjakan algoritma untuk masalah penugasan dalam graf bipartit ini, terlebih dahulu dengan cara menuliskan dualnya dalam program linier dari masalah penugasan. Variabel dualnya adalah u_i dengan *vertex-vertex* iϵS dan v_j dengan *vertex-vertex* jϵT.

Bentuk dualnya sebagai berikut:

Meminimalkan: $\sum u_i + \sum v_j$ (4.1.5)

Kendala: $u_i + v_j \geq c_{ij}$, untuk semua i, j (4.1.6)

$u_i, v_j \geq 0$, untuk semua i, j (4.1.7)

Dengan melengkapi variabel *slack* yang meliputi:

a) Jika $x_{ij} = 1$ maka $u_i + v_j = c_{ij}$

b) Jika $u_j \geq 0$ maka $\sum_j x_{ij} = 1$

c) Jika $v_j \geq 0$ maka $\sum_i x_{ij} = 1$

Algoritma Hungarian dimulai dengan solusi fisibel primal dan dual yang memenuhi kendala (a) dan (c) secara menyeluruh. Algoritma ini juga mencoba untuk mencari *path* perluasan dengan subgraf dibentuk oleh sisi-sisi yang variabel $u_i + v_j = c_{ij}$. Jika *path* ditemukan, maka *matching* baru menjadi fisibel. Jika *path* tidak dapat ditemukan, variabel-variabel dual diselesaikan sehingga sedikitnya satu sisi tambahan dapat menjadi subgraf tambahan.

4.2 Penyelesaian *matching* bobot maksimum masalah penugasan dalam graf bipartit dengan Algoritma Hungarian

Algoritma Hungarian mencoba mencari *path* perluasan, jika *path* ditemukan maka *matching* baru menjadi fisibel yaitu sisi-sisi yang memenuhi $u_i + v_j = c_{ij}$. Algoritma ini juga mencari adanya subgraf tambahan dari himpunan dual yang variabel-variabelnya u_i dengan *vertex-vertex* $i \in S$ dan v_j dengan *vertex-vertex* $j \in T$ yang diselesaikan.

Misal u_i = maksimal $_j[c_{ij}]$, $i \in S$; $v_j = 0$, $\pi_j = \infty$, $j \in T$.

Akan dikonstruksi subgraf terdiri atas semua sisi dengan $u_i + v_j = c_{ij.}$ Untuk setiap $i \in S$, pilih sisi pertama (i,j) sedemikian sehingga j *unmatched* dan

tempatkan pada himpunan *matching* M. Semua *vertex* belum diperiksa dan belum dilabel.

Langkah 1: Labeli setiap *vertex unmatched* i∈S dengan p(i) = 0.

Langkah 2: Pilih *vertex* belum diperiksa tetapi dilabeli i∈S atau j∈T, dan diberi nilai $\pi_j = 0$. Jika tidak ada ke langkah 5.

Langkah 3: Jika *vertex* yang dipilih dalam langkah 2 adalah i∈S, labeli *vertex* j∈T dengan p(j) = i (mengganti sembarang label yang ada) jika $u_i + v_j - c_{ij} < \pi_j$ dan gantilah π_j dengan $u_i + v_j - c_{ij}$. Jika *vertex* yang dipilih dalam langkah 2 adalah j∈T, kemudian tentukan label i jika j *unmatched*. Jika *vertex* j *unmatched*, ke langkah 4. Jika tidak, maka sisi (ij) di M dan beri label *vertex* i∈S dengan p(i) = j, kembali ke langkah 2.

Langkah 4: *Path* perluasan berakhir pada *vertex* i∈S atau j∈T yang akan dicari. Mencari *path* ini menggunakan fungsi *predecessor* p(). Perluasan *matching* dengan menambah sisi yang tidak berada di M dan menghapus sisi-sisi yang berada di M. Kita tulis $\pi_j = \infty$ $\forall\, j \in T$ dan hapus semua label dan kembali ke langkah 1.

Langkah 5: Dipilih $\delta_1 = \min \{u_i , i \in S\}$, $\delta_2 = \min \{\pi_j > 0 , j \in T\}$, dan $\delta = \min \{\delta_1, \delta_2\}$. Kita tulis kembali $u_i = u_i - \delta$ ∀ label *vertex* i∈S. Demikian juga $v_j = v_j + \delta$ ∀ j∈T dengan $\pi_j = 0$ dan $\pi_j = \pi_j - \delta$ ∀

vertex j∈T yang sudah dilabeli dengan $\pi_j > 0$. Jika $\delta = \delta_2$, kemudian ke langkah 2. Jika tidak, *matching* bobot maksimum telah ditemukan.

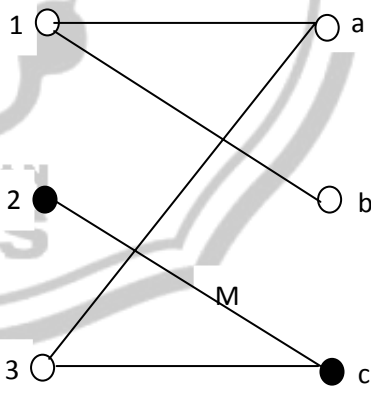
Untuk menggambarkan algoritma tersebut dapat dilihat dalam contoh di bawah ini. Notasikan *vertex* belum diperiksa dan *vertex* berlabel dengan L. Himpunan variabel dual dan *matching* diberikan sebagai berikut.

Contoh 4.2.1

Diketahui suatu graf bipartit H dengan himpunan variabel dual dan *matching* M di atas, dengan i ∈S untuk i = 1, 2, 3 dan j = a, b, c yang *matching* M = {(1, a)} sebagai berikut. Tentukan *matching* bobot maksimum dengan menggunakan algoritma Hungarian.

	a	b	c
1	20	19	18
2	16	12	20
3	18	16	18

Tabel 4.2.1

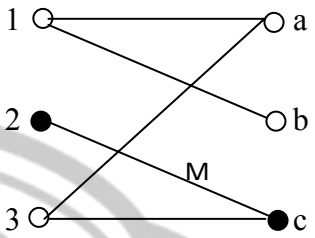


Gambar 4.2.1

Penyelesaian:

	a	b	c	u_i
1	20	19	18	20
2	16	12	20	20
3	18	16	18	18
v_j	0	0	0	
π_j	∞	∞	∞	

Tabel 4.2.1.a



Gambar 4.2.1.a

Langkah 1: *Vertex* 1 dan 3 *unmatched*. Beri label p(1) = 0, p(3) = 0.

Langkah 2: L = {1, 3}, pilih i = 1.

Langkah 3: Beri label *vertex* j∈T : p(a) = 1, p(b) = 1, p(c) = 1.

Vektor π baru: (0, 1, 2).

Langkah 2: L = {3, a}. Pilih i = 3.

Langkah 3: Beri label *vertex* j∈T : p(c) = 3.

Maka π = (0, 1, 0).

Langkah 2: L = {a}. Pilih j = a.

Langkah 3: *Vertex* a *unmatched*. Ke langkah 4.

Langkah 4: Perluasan *matching* dengan penelusuran balik label dari *vertex* i atau *vertex* j yang sudah diperiksa: p(a) = 1, p(1) = 0.

Dengan π = $(\infty, \infty, \infty, \infty)$ dan M = {(1, a), (2, c)}. Kembali ke langkah 1.

Langkah 1: *Vertex* 3 *unmatched*. Beri label p(3) = 0.

Langkah 2: L = {3}, pilih i = 3.

Langkah 3: Beri label *vertex* j∈T : p(a) = 3, p(b) = 3, p(c) = 3.

Vektor π baru: (0, 2, 0).

Langkah 2: L = {a}. Pilih j = a.

Langkah 3: *Vertex* a *matched*. Beri label p(1) = a.

Langkah 2: L = {1}. Pilih i = 1.

Langkah 3: Beri label *vertex* j∈T : p(b) = 1.

Vektor π baru: (0, 1, 0).

Langkah 2: L = {c}. Pilih j = c.

Langkah 3: *Vertex* c *matched*. Beri label p(2) = c.

Langkah 2: L = {2}. Pilih i = 2.

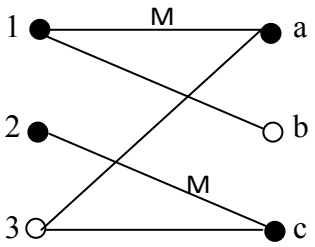
Langkah 3: Tidak ada label baru.

Langkah 2: Karena vertex *unmatched* tidak ada ke langkah 5.

Langkah 5: δ_1 = min {20, 20, 18} = 18; δ_2 = min {1} = 1; δ = min {18, 1} = 1. *Vertex* 1, 2 dan 3 telah diberi label. Pengambilan δ dari harga dual masing-masing *vertex*. Tambahkan δ pada v_a dan v_c serta kurangkan δ pada π_b. Maka hasilnya sebagai berikut.

	a	b	c	u_i
1	20	19	18	19
2	16	12	20	19
3	18	16	18	17
v_j	1	0	1	
π_j	0	0	0	

Tabel 4.2.1.b



Gambar 4.2.1.b

Langkah 2: L = {b}. Pilih j = b.

Langkah 3: *Vertex* b *unmatched*. Ke langkah 4.

Langkah 4: Perluasan *matching* dengan penelusuran balik label dari *vertex* i atau *vertex* j yang sudah diperiksa: p(b) = 3, p(3) = 0.

Dengan $\pi = (\infty, \infty, \infty, \infty)$ dan M = {(1, a), (2, c), (3, b)}. Kembali ke langkah 1.

Langkah 1: Tidak ada lagi *vertex* i $\in$ S yang bisa dipilih.

Langkah 2: Karena tidak ada *vertex unmatched*, ke langkah 5.

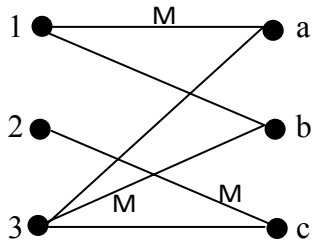
Langkah 5: δ_1 = min {19, 19, 17} = 17; δ_2 = min {∞} = ∞; $\delta \neq \delta_2$.

Karena $\delta \neq \delta_2$, maka *matching* bobot maksimum telah ditemukan M* = {(1, a), (2, c), (3, b)} dengan bobot dari *matching* bobot maksimum δ (M*) = 56.

Sehingga hasil akhirnya sebagai berikut.

	a	b	c	u_i
1	20	19	18	19
2	16	12	20	19
3	18	16	18	17
v_j	1	0	1	
π_j	0	0	0	

Tabel 4.2.1.c



Gambar 4.2.1.c

Contoh 4.2.2

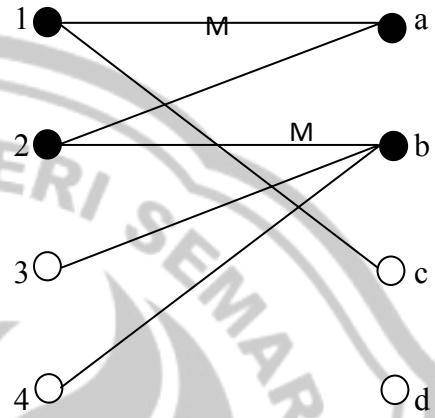
Sebuah toko mesin memiliki 4 buah mesin pengebor yang berbeda. Pada suatu hari tertentu datang 4 buah pekerjaan yang membutuhkan pengeboran. Upah yang harus diberikan pemilik toko pada para pekerja-pekerjanya perhari perorangnya (dalam ribuan rupiah) direpresentasikan dalam bentuk matriks $[c_{ij}]$, dengan $c_{ij} = \delta\ (ij)$ untuk $i \leq 4$ dan $j \leq 4$ yang merupakan penggambaran matriks dari graf bipartit.

Tentukan *matching* bobot maksimum dan biaya/upah minimum dengan menggunakan algoritma Hungarian (upah pekerja dalam satuan ribu rupiah).

Diberikan juga suatu graf bipartit G dengan himpunan variabel dual dan *matching* M di atas, dengan i ∈S untuk i = 1, 2, 3, 4 dan j = a, b, c, d yang *matching* M = {(1, a), (2, b)} sebagai berikut.

	a	b	c	d
1	32	18	32	26
2	22	24	12	16
3	24	30	26	24
4	26	30	28	20

Tabel 4.2.2

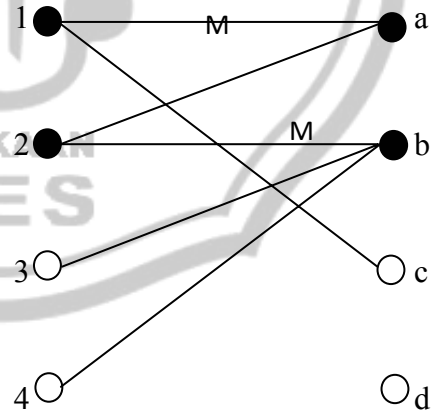


Gambar 4.2.2

Penyelesaian:

	a	b	c	d	u_i
1	32	18	32	26	32
2	22	24	12	16	24
3	24	30	26	24	30
4	26	30	28	20	30
v_j	0	0	0	0	
π_j	∞	∞	∞	∞	

Tabel 4.2.2.a



Gambar 4.2.2.a

Langkah 1: *Vertex* 3 dan 4 *unmatched*. Beri label p(3) = 0, p(4) = 0.

Langkah 2: L = {3, 4}, pilih i = 3.

Langkah 3: Beri label *vertex* j∈T : p(a) = 3, p(b) = 3, p(c) = 3, p(d) = 3.

Vektor π baru: (6, 0, 4, 6).

Langkah 2: L = {4, b}. Pilih i = 4.

Langkah 3: Beri label *vertex* j∈T : p(a) = 4, p(c) = 4.

Maka π = (4, 0, 2, 6).

Langkah 2: L = {b}. Pilih j = b.

Langkah 3: *Vertex* b *matched*. Beri label *vertex* 2, p(2) = b.

Langkah 2: L = {2}. Pilih i = 2.

Langkah 3: Beri label *vertex* j∈T : p(a) = 2.

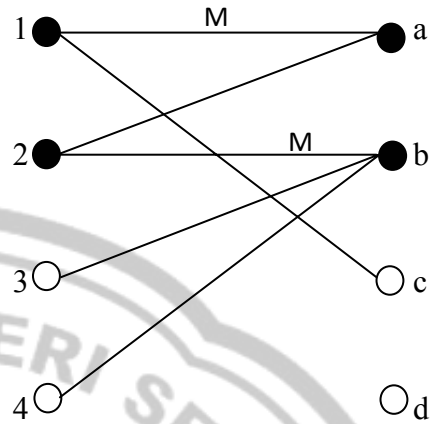
Maka π = (2, 0, 2, 6).

Langkah 2: Karena *vertex* yang dipilih tidak ada, ke langkah 5.

Langkah 5: δ_1 = min {32, 24, 30, 30} = 24; δ_2 = min {2, 2, 6} = 2; δ = min {24, 2} = 2. *Vertex* 2, 3 dan 4 diberi label. Pengambilan δ dari harga dual dari masing-masing *vertex*. Tambahkan δ pada v_b dan kurangkan δ pada π_a, π_c, π_d. Maka hasilnya menjadi:

	a	b	c	d	u_i
1	32	18	32	26	32
2	22	24	12	16	22
3	24	30	26	24	28
4	26	30	28	20	28
v_j	0	2	0	0	
π_j	0	0	0	4	

Tabel 4.2.2.b



Gambar 4.2.2.b

Langkah 2: L = {c}. Pilih j = c.

Langkah 3: *Vertex* c *unmatched* ke langkah 4.

Langkah 4: Perluasan *matching* dengan penelusuran balik label dari *vertex* i atau *vertex* j yang sudah diperiksa: p(c) = 4, p(4) = 0.

Dengan $\pi = (\infty, \infty, \infty, \infty)$ dan M = {(1, a), (2, b), (4, c)}. Kembali ke langkah 1.

Langkah 1: *Vertex* 3 *unmatched*. Beri label p(3) = 0.

Langkah 2: L = {3}. Pilih i = {3}.

Langkah 3: Beri label p(a) = 3, p(b) = 3, p(c) = 3, p(d) = 3. Maka π = (4, 0, 2, 4).

Langkah 2: L = {b}. Pilih j = b.

Langkah 3: *Vertex* b *matched*. Beri label p(2) = b.

Langkah 2: L = {2}. Pilih i = 2.

Langkah 3: Beri label p(a) = 2. Maka π = (0, 0, 2, 4).

Langkah 2: L = {a}. Pilih j = a.

Langkah 3: *Vertex* a *matched*. Beri label p(1) = a.

Langkah 2: L = {l}. Pilih i = 1.

Langkah 3: Beri label p(c) = 1. Maka π = (0, 0, 0, 4).

Langkah 2: L = {c}. Pilih j = c.

Langkah 3: *Vertex* c *matched*. Beri label p(4) = c.

Langkah 2: L = {4}. Pilih i = 4.

Langkah 3: Tidak ada label baru.

Langkah 2: Karena *vertex unmatched* tidak ada, ke langkah 5.

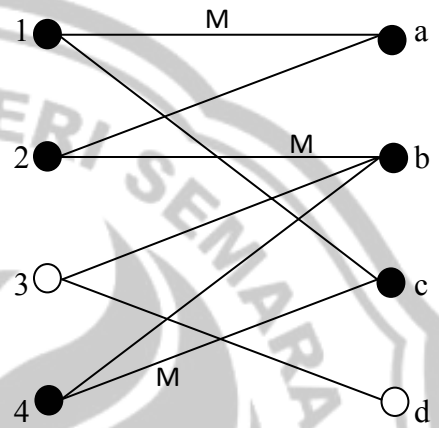
Langkah 5: δ_1 = min {32, 22, 28, 28} = 22; δ_2 = min {4} = 4; δ = min {22, 4} = 4.

Vertex 1, 2, 3 dan 4 telah diberi label. Pengambilan δ dari harga dual

masing-masing *vertex*. Tambahkan δ pada v_a, v_b dan v_c serta kurangkan δ pada π_d. Maka hasilnya sebagai berikut.

	a	b	c	d	u_i
1	32	18	32	26	28
2	22	24	12	16	18
3	24	30	26	24	24
4	26	30	28	20	24
v_j	4	6	4	0	
π_j	0	0	0	0	

Tabel 4.2.2.c



Gambar 4.2.2.c

Langkah 2: L = {d}. Pilih j = d.

Langkah 3: *Vertex* d *unmatched*. Ke langkah 4.

Langkah 4: Perluasan *matching* dengan penelusuran balik label dari *vertex* i atau *vertex* j yang sudah diperiksa: p(d) = 3, p(3) = 0.

Dengan $\pi = (\infty, \infty, \infty, \infty)$ dan M = {(1, a), (2, b), (3, d), (4, c)}. Kembali ke langkah 1.

Langkah 1: Tidak ada lagi *vertex* i ∈ S yang bisa dipilih.

Langkah 2: Karena tidak ada *vertex unmatched*, ke langkah 5.

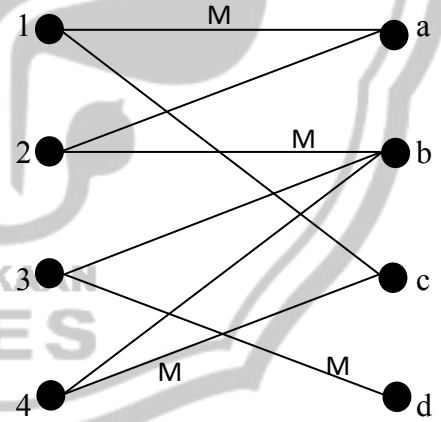
Langkah 5: δ_1 = min {28, 18, 24, 24} = 18; δ_2 = min {∞} = ∞; $\delta \neq \delta_2$.

Karena $\delta \neq \delta_2$, maka *matching* bobot maksimum telah ditemukan M* = {(1, a), (2, b), (3, d), (4, c)} dengan bobot dari *matching* bobot maksimum δ (M*) = 108. Sedangkan biaya/ upah minimum pekerja perhari adalah δ_1 = 18.

Jadi dapat disimpulkan bahwa biaya/ upah minimum yang harus dibayar pemilik toko mesin kepada para pekerjanya sebesar Rp 18.000,00 perhari perorangnya pada 4 buah mesin dengan 4 buah pekerjaan untuk setiap mesin hanya melakukan satu pekerjaan. Sehingga hasil akhirnya sebagai berikut.

	a	b	c	d	u_i
1	32	18	32	26	28
2	22	24	12	16	18
3	24	30	26	24	24
4	26	30	28	20	24
v_j	4	6	4	0	
π_j	0	0	0	0	

Tabel 4.2.2.d



Gambar 4.2.2.d

BAB V

SIMPULAN DAN SARAN

5.1 Simpulan

Langkah-langkah yang digunakan untuk menyelesaikan *matching* bobot maksimum masalah penugasan sebagai berikut.

1. Memodelkan permasalahan tersebut dengan program linier sebagai berikut.

   Memaksimalkan: $\sum_i \sum_j c_{ij} x_{ij}$

   Kendala: $\sum_j x_{ij} \leq 1$, untuk semua i $\in$ S

   $\sum_i x_{ij} \leq 1$, untuk semua j $\in$ T

   $x_{ij} \geq 0$, untuk semua i $\in$ S, j $\in$ T

2. Menyelesaikan bentuk dual dari program linier pada langkah 1.

   Bentuk dualnya sebagai berikut.

   Meminimalkan: $\sum u_i + \sum v_j$

   Kendala: $u_i + v_j \geq c_{ij}$, untuk semua i, j

   $u_i, v_j \geq 0$, untuk semua i, j

3. Menyelesaikan model dual tersebut dengan algoritma Hungarian.

   Misal u_i = maksimal$_j[c_{ij}]$, i $\in$ S; $v_j = 0$, $\pi_j = \infty$, j $\in$ T.

   Akan dikonstruksi subgraf terdiri atas semua sisi dengan $u_i + v_j = c_{ij}$. Untuk setiap i $\in$ S, pilih sisi pertama (i,j) sedemikian sehingga j *unmatched* dan tempatkan pada himpunan *matching* M. Semua *vertex* belum diperiksa dan belum dilabel.

   Langkah 1: Labeli setiap *vertex unmatched* i$\in$S dengan p(i) = 0.

Langkah 2: Pilih *vertex* belum diperiksa tetapi dilabeli i∈S atau j∈T, dan diberi nilai π_j = 0. Jika tidak ada ke langkah 5.

Langkah 3: Jika *vertex* yang dipilih dalam langkah 2 adalah i∈S, labeli *vertex* j∈T dengan p(j) = i (mengganti sembarang label yang ada) jika $u_i + v_j - c_{ij} < \pi_j$ dan gantilah π_j dengan $u_i + v_j - c_{ij}$. Jika *vertex* yang dipilih dalam langkah 2 adalah j∈T, kemudian tentukan label i jika j *unmatched*. Jika *vertex* j *unmatched*, ke langkah 4. Jika tidak, maka sisi (ij) di M dan beri label *vertex* i∈S dengan p(i) = j, kembali ke langkah 2.

Langkah 4: *Path* perluasan berakhir pada *vertex* i∈S atau j∈T yang akan dicari. Mencari *path* ini menggunakan fungsi *predecessor* p(). Perluasan *matching* dengan menambah sisi yang tidak berada di M dan menghapus sisi-sisi yang berada di M. Kita tulis $\pi_j = \infty \ \forall j \in T$ dan hapus semua label dan kembali ke langkah 1.

Langkah 5: Dipilih δ_1= min $\{u_i , i \in S\}$, δ_2= min $\{\pi_j > 0 , j \in T\}$, dan δ = min $\{\delta_1, \delta_2\}$. Kita tulis kembali $u_i = u_i - \delta \ \forall$ label *vertex* i∈S. Demikian juga $v_j = v_j + \delta \ \forall$ j∈T dengan π_j = 0 dan $\pi_j = \pi_j - \delta \ \forall$ *vertex* j∈T yang sudah dilabeli dengan $\pi_j > 0$. Jika $\delta = \delta_2$, kemudian ke langkah 2. Jika tidak, *matching* bobot maksimum telah ditemukan.

5.2 Saran

Berkaitan dengan hasil penelitian, ada beberapa hal yang perlu mendapat perhatian yaitu penelitian ini hanya mengkaji penyelesaian *matching* bobot maksimum masalah penugasan dalam graf bipartit. Untuk itu

perlu penelitian lebih lanjut tentang algoritma lain untuk jenis graf bipartit lengkap.

DAFTAR PUSTAKA

Budayasa, I Ketut. 1997. *Matematika Diskrit I.* Surabaya: Balai Pustaka.

Chartrand,Gary and Oellermann, O.R. 1993. *Applied and Algorithmic Graph Theory*. New York: Mc Graw Hill Inc.

Dimyati, Tjutju Tarliah. 1987. *Operation Research*. Bandung: Sinar Baru Algensindo.

Evans, J. R and Minieka, E. 1992. *Optimization Algorithms for Network and Graph*. New York: Marcell Dekker Inc.

Goodaire, Edgar. G and Parmenter, M. M. 1998. *Discrete Mathematics With Graph Theory*. United States of America: Prentice Hall, Inc.

Hariyadi, Charles. *Eksentrik Digraf dari Graf Star, Graf Double Star, dan Graf Komplit Bipartit*. Tersedia di: http://mail.informatika.org/~rinaldi/Matdis/2006-2007/Makalah/Makalah0607-60.pdf [5 Februari 2009].

Munir, Rinaldi. 2001. *Buku Teks Ilmu Komputer Matematika Diskrit*. Bandung: Informatika Bandung.

Rompah, Edward. *.Algoritma Hungaria*. Tersedia di: http://edwardgr.wordpress.com/2009/02/03/algoritma-hungaria/ [3 Februari 2009].

Rosen, Kenneth. H. 2003. *Discrete Mathematics and Its Applications*. New York: Mc Graw Hill Inc.

Siang, Jong Jek. 2002. *Matematika Diskrit dan Aplikasinya pada Ilmu Komputer*. Yogyakarta: Andi.

Susanto, Alvin. Penggunaan Algoritma Hungarian Dalam Menyelesaikan Persoalan Matriks Berbobot. Tersedia di: http://www.informatika.org/~rinaldi/Stmik/2007-2008/Makalah2008/MakalahIF2251-2008-074.pdf [5 Februari 2009].

Sutarno, Heri. 2003. *Matematika Diskrit*. Bandung: JICA.