BAB II

LANDASAN TEORI

2.1 Graf

2.1.1 Definisi Graf

Graf G merupakan representasi dari suatu masalah yang digambarkan sebagai sekumpulan noktah (simpul) yang dihubungkan dengan sekumpulan garis (sisi). Graf didefinisikan sebagai himpunan (V,E) yang dalam hal ini:

V = himpunan berhingga dan tidak kosong dari simpul – simpul (vertices atau node) = { v_1, v_2,…..,v_n}, dan

E = himpunan sisi (edges atau arcs) yang menghubungkan sepasang simpul = { e_1, e_2,….., e_n } atau dapat ditulis singkat G = (V, E).

Simpul pada graf dapat dinomori dengan huruf, seperti v, w,….., dengan bilangan asli 1, 2, 3,... atau dengan gabungan keduanya. Sedangkan sisi yang menghubungkan simpul vi dengan simpul vj dinyatakan dengan pasangan (v_i, v_j) atau dengan lambang e_1, e_2,... Dengan kata lain jika e adalah sisi yang menghubungkan simpul v_i dengan simpul v_j, maka e dapat ditulis sebagai e = (v_i,v_j).

2.1.2 Jenis – jenis Graf

Graf dapat dikelompokkan menjadi beberapa jenis bergantung pada sudut pandang pengelompoknya. Pengelompokkan graf dapat dipandang berdasarkan ada tidaknya sisi ganda atau sisi kalang. Berdasarkan jumlah simpul, atau berdasarkan orientasi arah pada sisi.

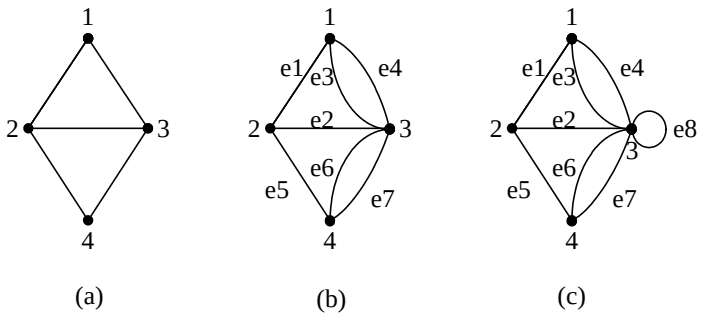
Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, maka secara umum graf digolongkan menjadi dua jenis:

a. Graf Sederhana (*simple graph*)

   Graf yang tidak mengandung gelang maupun sisi ganda dinamakan graf sederhana, seperti terdapat pada gambar 2.1(a)

b. Graf Tak sederhana (*unsimple graph*)

   Graf yang mengandung sisi ganda atau gelang dinamakan graf tak sederhana (*unsimple graph*). Ada dua macam graf tak sederhana, yaitu graf ganda (*multi graph*) dan graf semu (*pseudograph*). Graf ganda adalah graf yang mengandung sisi ganda, seperti terdapat pada gambar 2.1 (b). sisi ganda dapat diandaikan sebagai saluran telepon tambahan apabila beban komunikasi data antar komputer sangat padat. Graf semu lebih umum daripada graf ganda, karena sisi pada graf semu dapat terhubung ke dirinya sendiri, seperti terdapat pada gambar 2.1(c) dibawah ini.



Gambar 2.1 Tiga buah graf (a) graf sederhana, (b) graf ganda, (c)graf semu

2.1.3 Terminologi Graf

Terminologi (istilah) pada pembahasan mengenai graf cukup banyak. Terminologi yang sering dipakai meliputi :

a. Ketetanggaan (*Adjacent*)

   Dua buah simpul dikatakan bertetanggaan bila keduanya terhubung langsung. Secara formal dinyatakan :

   V_j bertetangga dengan v_k jika sedemikian sehingga e = (v_j, v_k) pada gamba 2.2 (a), simpul 1 bertetangga dengan simpul 2 dan 3, tetapi tidak bertetangga dengan simpul 4.

b. Derajat (*Degree*)

   Derajat suatu simpul adalah jumlah sisi yang bersisian dengan simpul tersebut. Notasi : d (v)

   Simpul yang mempunyai gelang dihitung mempunyai dua buah sisi yang bersisian dengannya. Jadi untuk graf pada gambar 2.2(b), d(2)=4.

c. Lintasan (*Path*)

Lintasan dari simpul vp ke vq dalam G ialah rangkaian simpul v_p, v_{i1}, v_{i2}, …v_{im}, v_q sehingga (v_p, v_{i1}), (v_{i1}, v_{i2}),…,(v_{im}, v_q) adalah sisi – sisi dalam graf G. Pada gambar 2.2 (a), lintasan 1, 2, 4, 3 adalah lintasan dengan barisan sisi (1,2), (2,4), (4,3). Simpul yang dilalui boleh berulang.

d. Sirkuit (*Cirkuit*)

Lintasan elementer (lintasan dengan semua simpul yang dilalui hanya muncul satu kali) dengan simpul terakhir disebut sirkuit. Pada gambar 2.2(a) 1, 2, 3, 1 adalah sebuah sirkuit. Panjang sirkuit adalah jumlah sisi dalam sirkuit tersebut. Lintasan 1, 2, 3, 1 pada gambar 2.2(a) memiliki panjang 3. Sirkuit sederhanan adalah sirkuit dengan semua sisi yang dilalui hanya satu kali. Sirkuit elementer adalah sirkuit dengan semua simpul yang dilalui hanya satu kali.

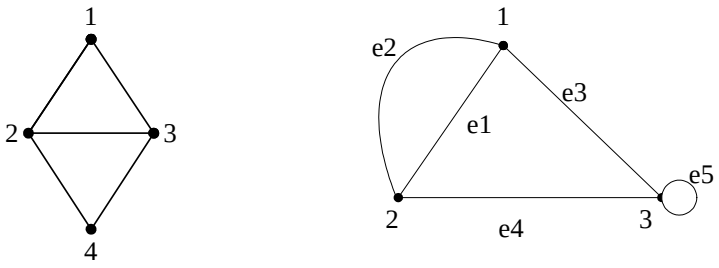
e. Terhubung (*Connected*)

Dua buah simpul v_1 dan v_2 dikatakan terhubung jika terdapat lintasan dari v_1 ke v_2. Graf tak berarah G disebut graf terhubung jika untuk setiap pasang simpul v_i dan v_j dalam himpunan V terdapat lintasan dari v_i ke v_j. Jika tidak, maka G disebut graf tak berhubung.

f. Graf Berbobot (*Weighted Graph*)

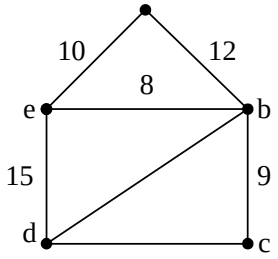
Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada setiap sisi dapat menyatakan jarak antara 2 buah kota, biaya perjalanan antara dua buah kota, waktu tempuh pesan

(message) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya. Gambar 2.3 adalah contoh graf berbobot.



(a) G_1 (b) G_2

Gambar 2.2 Dua buah graf G_1 dan G_2



Gambar 2.3 Graf berbobot

2.2 Basis Data

Pada bagian ini akan dibahas mengenai definisi basis data, operasi dasar basis data, model data yang dipergunakan dalam basis data serta model keterhubungan antar entitas (*entity relationship*/ER)

2.2.1 Definisi Basis Data

Basis data terdiri atas dua kata [1], yaitu *basis* dan *data*. *Basis* kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan *data* adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, kedaan, dan sebagainya, yang direkam dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya.

Basis data dapat didefinisiskan dalam sejumlah sudut pandang seperti [1]:

- Himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan dengan cepat dan mudah.
- Kumpulan data yang saling berhubungan yang disimpan secara bersama sedemikian rupa dan tanpa pengulangan (redudansi) yang tidak perlu, untuk memenuhi berbagai kebutuhan.
- Kumpulan file/tabel/arsip yang saling berhubungan yang disimpan dalam media penyimpanan elektronik.

2.2.2 Operasi Dasar Basis Data

Didalam sebuah *disk*, basis data dapat diciptakan dan dapat pula ditiadakan. Dalam sebuah basis data, kita dapat menempatkan satu atau lebih file/tabel. Pada tabel inilah sesungguhnya data disimpan/ditempatkan. Setiap basis data umumnya dibuat untuk mewakili sebuah semesta data yang spesifik, misalnya basis data kepegawaian, basis data akademik, basis data inventori (pergudangan), dan sebagainya. Sementara dalam basis data akademik, kita dapat menempatkan file

mahasiswa, file mata_kuliah, file dosen, file jadwal, file kehadiran, file nilai dan seterusnya.

Operasi-operasi yang dapat dilakukan berkenaan dengan basis data dapat meliputi:

- Pembuatan basis data baru (*create databse*).
- Penghapusan basis data (*drop database*).
- Pembuatan file/tabel baru (*create table*).
- Penghapusan file/tabel dari suatu basis data (*drop table*).
- Penambahan/pengisian data baru ke sebuah file/tabel pada sebuah basis data (*insert*).
- Menampilkan data dari sebuah file/tabel (*select*).
- Pengambilan data dari sebuah file/tabel (*retrieve*/*search*).
- Pengubahan data dari sebuah file/tabel (*update*)
- Penghapusan data dari sebuah file/tabel (*delete*).

2.2.3 Model Data

Model data dapat didefinisikan sebagai kumpulan perangkat konseptual untuk menggambarkan data, hubungan data, semantik (makna) data dan batasan data. Karena itu yang ingin ditunjukan adalah makna dari data dan keterhubungan dengan data lain, maka model data ini lebih tepat jika disebut “Model Data Logic”. Penyingkatan menjadi model data dilakukan demi penyederhanaan penulisan.

Ada sejumlah cara dalam merepresentasikan model data dalam perancangan basis data, yang secara umum dapat dibagi dalam 2 (dua) kelompok , yaitu:

1. Model Data berdasarkan objek, yang terdiri dari:
   - Model keterhubungan entitas (*Entity-Relationship Model*)
   - Model berorientasi objek (*Object-Oriented Model*)
   - Model data semantik (*Semantic Data Model*)
   - Model data fungsional (*Functional Data Model*)
2. Model Data berdasarkan record, yang terdiri dari:
   - Model Relasional (*Relational Model*)
   - Model Hirarkis (Hierarchical Model)
   - Model Jaringan (*Network Model*)

2.2.4 Model Keterhubungan Entitas

Pada model *Entity-Relationship*, semesta data yang ada di "dunia nyata" diterjemahkan atau ditransformasikan dengan memanfaatkan sejumlah perangkat konseptual menjadi sebuah diagram data, yang umum disebut diagram *Entity-Relationship*. Terdapat dua komponen utama pembentuk model *entity relationship* yaitu entitas (*entity*) dan relasi (*relation*), kedua komponen ini akan dideskripsikan lebih lanjut melalui sejumlah atribut/properti.

Entitas merupakan sebuah individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Sekelompok entitas yang sejenis dan berada dalam lingkup yang sama membentuk sebuah himpunan entitas (*entity set*) akan tetapi pada beberapa literatur himpunan entitas disebutkan

sebagai entitas saja. Sederhananya entitas menunjuk pada individu suatu objek, sedangkan himpunan entitas menunjuk pada famili dari individu tersebut. Perbedaan antara entitas dan himpunan entitas dapat terlihat dari beberapa contoh berikut, sebagai himpunan entitas adalah semua pelanggan atau pelanggan saja, sedangkan entitasnya adalah Dicky, Alink, Ivan dan sebagainya.

Setiap entitas memiliki atribut yang mendeskripsikan karakteristik (properti) dari entitas tersebut. Penentuan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting dalam pembentukan model data. Penetapan atribut bagi sebuah entitas umunnya memang didasarkan pada fakta yang ada, tetapi tidak selalu seperti itu. Adapun jenis-jenis atribut adalah sebagai berikut:

- *Key* dan Atribut Deskriptif

  Key adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris data (*row*) dalam tabel secara unik, artinya jika suatu atribut dijadikan sebagai *key*, maka tidak boleh ada dua atau lebih baris data dengan nilai yang sama untuk atribut tersebut. Sedangkan atribut yang tidak termasuk sebagai *key* disebut atribut deskriptif.

- Atribut Sederhana dan Atribut Komposit

  Atribut sederhana adalah atribut atomik yang tidak dapat dipilih lagi, sementara atribut komposit merupakan atribut yang masih dapat diuraikan lagi menjadi sub-sub atribut yang masing-masing atribut memiliki makna.

- Atribut Bernilai Tunggal dan Atribut Bernilai Banyak

Atribut yang bernilai tunggal ditujukan pada atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data, sedangkan atribut bernilai banyak ditunjukan pada atribut-atribut yang dapat kita isi dengan lebih dari satu nilai, tetapi jenisnya sama.

- Atribut Harus Bernilai (*Mandatory Attribute*) dan Nilai Null

  Ada sejumlah atribut pada sebuah tabel yang ditetapkan harus berisi data. Jadi nilainya tidak boeh kosong. Atribut atribut semacam ini disebut *Mandatory Attribute* sebaliknya terdapat pula atribut-atribut yang pada suatu tabel yang nilainya boleh dikosongkan.yang disebut *Non Mandatory Attribute*, nilai (konstanta) Null digunakan untuk mengisi atribut-atribut yang nilainya belum ada atau tidak ada.

- Atribut Turunan

  Atribut turunan adalah atribut yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut atau tabel lain yang berhubungan. Atribut demikian sebenarnya dapat dihilangkan dari sebuah tabel karena nilai-nilainya bergantung pada nilai yang ada di atribut lain.

Relasi menunjukan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Misalnya, entitas seorang mahasiswa dengan NIM=’10101157’ dan nama_mahasiswa=’Rusdiyanto’ (yang ada pada himpunan entitas Mahasiswa) mempunyai relasi dengan entitas sebuah mata kuliah dengan kode_kuliah=’IF32210’ dan nama_kuliah=’Kalkulus II’. Relasi diantara kedua entitas tadi mengandung arti bahwa mahasiswa (Rusdiyanto/10101157) sedang mengambil mata kuliah (IF32210/Kalkulus II).

Dalam model *Entity Relationship* dikenal pula nilai kardinalitas, kardinalitas atau derajat relasi adalah nilai yang menunjukan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Derajat relasi yang diantara dua himpunan entitas (misalnya A dan B) dapat berupa:

- Satu ke Satu (*One to One*)

  Ini berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B, dan begitu juga sebaliknya.

- Satu ke Banyak (*One To Many*)

  Misalkan direpresentasikan dalam bentuk pencantuman atribut *key* dari himpunan entitas A (berderajat 1) ke himpunan entitas B (berderajat N), ini berarti setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya.

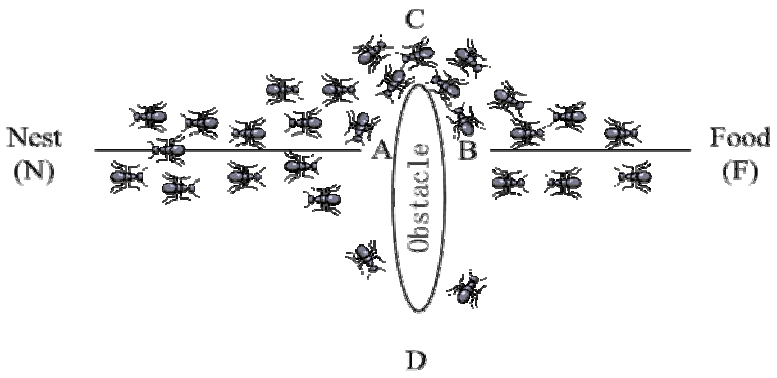
- Banyak ke Satu (*Many to One*)

  Misalkan direpresentasikan dalam bentuk pencantuman atribut *key* dari entitas B (berderajat 1) ke himpunan himpunan entitas A (berderajat N), ini berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya.

- Banyak ke Banyak (*many to many*)

2.3 Algoritma M*ax-Min Ant System*

Ant Colony Optimization (ACO) pertama kali diperkenalkan oleh Marco Dorigo, ACO itu sendiri terinspirasi oleh koloni-koloni semut dalam mencari makan. Semut-semut tersebut meninggalkan zat (*pheromone*) di jalan yang mereka lalui. Algoritma ACO ini merupakan algoritma pencarian berdasarkan probabilistik, di mana probabilistik yang digunakan merupakan probabilistik dengan bobot sehingga butir pencarian dengan bobot yang lebih besar akan berakibat memiliki kemungkinan terpilih yang lebih besar pula.



Gambar 2.4 Perilaku nyata semut ketika mencari makan

Dari gambar di atas terlihat dalam perjalanannya, semut-semut dari sarang menuju ke tempat makanan ketika menemui hambatan maka semut-semut tersebut akan memilih jalan yang berbeda, jalur A-C-B merupakan jalur yang paling banyak dilalui semut daripada jalur A-D-B sehingga kandungan *pheromone* A-C-B lebih besar daripada kandungan *pheromone* A-D-B, dengan demikian alur A-C-B akan memiliki kemungkinan untuk dilalui kembali.

Ant Colony Optimation itu sendiri dapat diterapkan untuk menyelesaikan berbagai macam masalah diantaranya *Traveling Salesman Problem*, *Minimum*

Spanning Tree Problem, *Scheduling Problems* dan lain-lain. Algoritma umum dari ACO adalah sebagai berikut:

```
masukan data permasalahan
while not selesai do
     bangkitkan semut
     bangkitkan solusi
     update pheromone
     hancurkan semut
end while
keluarkan solusi
```

Gambar 2.5 Algoritma umum ACO

Algoritma ACO terdiri atas dua versi besar yaitu:

1. ACS (Ant Colony System)
2. MMAS (*MAX-MIN Ant System*)

Perbedaan umum dari kedua versi itu adalah pada bagian *update pheromone*. Dan pada fungsi probabilistik ACS menggunakan fungsi heuristik sedangkan MMAS tidak menggunakan fungsi heuristik. Selain itu, pada MMAS terdapat parameter *pheromone maximum* dan *pheromone minimum* yang tidak diterapkan pada ACS. Alasan penggunaan batasan maksimum dan minimum tersebut adalah untuk memberikan keseimbangan level intensifikasi dan diversifikasi pencarian. Penggunaan nilai maksimum yang terlalu tinggi akan berakibat semua semut menghasilkan solusi yang sama sehingga algoritma menjadi stagnan. Penggunaan nilai minimum yang terlalu rendah bahkan sama dengan nol akan berakibat jalur tersebut semakin tidak berpeluang untuk dipilih sehingga akan membatasi ruang solusi.

Perbedaan dari ACS dan MMAS dapat terlihat lebih jelas pada persamaan *update pheromone* berikut ini:

1. ACS

- Persamaan probabilistik ACS:

$$p_{e_i,t}(\tau(A_{i-1}),\eta(A_{i-1})) = \frac{(\tau_{(e_i,t)}(A_{i-1}))^{\alpha} \cdot (\eta_{(e_i,t)}(A_{i-1}))^{\beta}}{\sum_{\theta \in T}(\tau_{(e_i,\theta)}(A_{i-1}))^{\alpha} \cdot (\eta_{(e_i,\theta)}(A_{i-1}))^{\beta}}$$

-Persamaan *update pheromone* ACS lokal:

$$\tau_{(e_i,t_{chosen})} \leftarrow (1-\alpha) \cdot \tau_{(e_i,t_{chosen})} + \alpha \cdot \tau_0$$

- Persamaan *update pheromone* ACS global:

$$\tau_{(e,t)} \leftarrow \begin{cases} (1-\rho).\tau_{(e,t)} + \rho.\dfrac{g}{1+q(C_{global_best})} & \text{, jika (e,t) adalah } C_{global_best} \\ (1-\rho).\tau_{(e,t)} & \text{, jika (e,t) bukan } C_{global_best} \end{cases}$$

2. MMAS

- Persamaan untuk menghitung nilai probabilistik:

$$p_{(e,t)} = \frac{\tau_{(e,t)}}{\sum_{i=0}^{|T|} \tau_{(e,t_i)}}$$

- Persamaan untuk *update pheromone*:

$$\tau_{(e,t)} \leftarrow \begin{cases} (1-\rho).\tau_{(e,t)} + 1 & \text{, jika (e,t) adalah } C_{global_best} \\ (1-\rho).\tau_{(e,t)} & \text{, jika (e,t) bukan } C_{global_best} \end{cases}$$

$$\tau_{(e,t)} \leftarrow \begin{cases} \tau_{\min}, jika\, \tau_{(e,t)} < \tau_{\min} \\ \tau_{\max}, jika\, \tau_{(e,t)} > \tau_{\max} \\ \tau_{(e,t)} \end{cases}$$

Dalam peghitungan nilai probabilistik untuk setiap kemungkinan pada ACS menggunakan nilai heuristik (η) sedangkan pada *MMAS* heuristik tersebut dihilangkan. Selain hal tersebut perbedaan antara ACS dan MMAS dapat terlihat dari persamaan *update pheromone*, pada ACS *update pheromone* dilakukan dua

kali yaitu setiap semut melewati node (*update pheromone* lokal) dan setiap satu koloni semut melewati node (*update pheromone* global), sedangkan pada *MMAS* hanya dilakukan *update pheromone* global, tetapi pada *MMAS* digunakan parameter T_{max} (*pheromone* maksimum) dan T_{min} (*pheromone* minimum).

Tabel 2.1 Parameter yang digunakan dalam ACO

Nama parameter	Keterangan
$\tau_{(e,t)}$	*Pheromone* dari event *e* dan *timeslot t*
ρ	Derajat penguapan *pheromone*
P(e,t)	Probabilitas dari *event* dan *timeslot*
g	Faktor skala
$q(C_{global_best})$	Nilai kecocokan solusi dari solusi terbaik global
τ_{min}	*Pheromone* minimum
τ_{max}	*Pheromone* maksimum

2.4 Model Matematis Graf sebagai Representasi *Hard Constraint*

Constraint adalah kendala yang didefinisikan agar penjadwalan yang dihasilkan merupakan jadwal yang baik, constraint itu sendiri secara umum dibagi menjadi dua kategori yaitu: *soft* (lembut) dan *hard* (keras). *Soft constraint* adalah kendala-kendala yang memungkinkan untuk dilanggar karena pelanggaran terhadap *soft constraint* tidak akan mempengaruhi kebenaran hasil penjadwalan, namun jika terjadi pelanggaran terhadap *soft constraint* ini maka jadwal yang dihasilkan bukan merupakan hasil yang nyaman untuk digunakan. Sedangkan *hard constraint* adalah kendala di mana tidak boleh terjadi pelanggaran terhadap

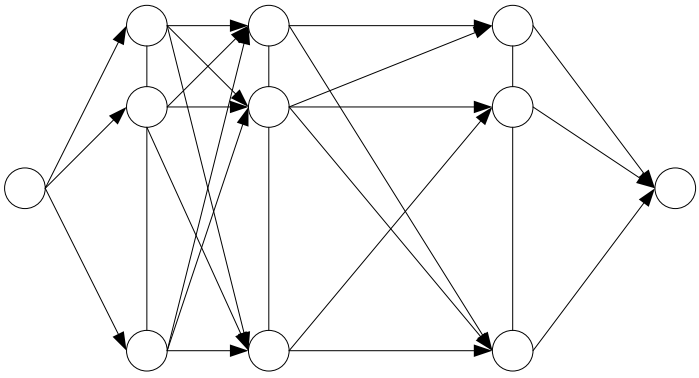
kendala-kendala yang telah ditetapkan. Untuk merepresentasikan *hard constraint* yang ada dibangunlah sebuah model matematis dalam bentuk graf.

Graf G didefinisikan sebagai pasangan himpunan (V,E), di mana :

V = himpunan berhingga yang tidak kosong dari simpul-simpul atau *vertex*/node $\{V_1, V_2, \ldots, V_n\}$.

E = himpunan sisi (*edge* atau *arc*) yang menghubungkan sepasang simpul $\{ e_1, e_2, \ldots, e_n\}$.

Untuk masalah penjadwalan perkuliahan yang akan diselesaikan dengan menggunakan algoritma *MMAS* ini, graf yang dibangun merupakan graf berarah. Node-node yang terbentuk nantinya merupakan representasi dari event-event yang akan dijadwalkan pada waktu *t*, di mana event-event tersebut merupakan sekumpulan data yang terdiri dari kelas, jenjang, semester dan mata kuliah yang diikuti. Sedangkan keterhubungan antar node-node yang terbentuk merupakan representasi dari *hard constraint* yang ada, sebagai contoh dapat dilihat pada gambar di bawah ini.



Gambar 2.6 Graf berarah yang merupakan pasangan *event* dan *timeslot*