

Desain dan Analisis Algoritma Penyelesaian Permasalahan Penugasan Bersyarat dengan Representasi *Bipartite Graph*

Muhammad Izzuddin, Arya Yudhi Wijaya, Rully Soelaiman

Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)

Jl. Arief Rahman Hakim, Surabaya 60111 Indonesia

e-mail: rully@is.its.ac.id

Abstrak—Permasalahan dalam penelitian ini adalah permasalahan penugasan bersyarat dimana terdapat beberapa pekerjaan dan beberapa orang pekerja. Setiap pekerjaan harus dilakukan oleh semua orang yang ada serta setiap orang memiliki waktu yang dibutuhkan tersendiri dalam menyelesaikan pekerjaan tersebut. Setiap orang hanya dapat mengerjakan sebuah pekerjaan dan sebuah pekerjaan hanya dapat dikerjakan oleh satu orang dalam satu waktu. Pekerjaan yang dimaksud juga bersifat independen dalam artian dapat dilakukan kapanpun oleh setiap orang. Semua orang juga dapat berhenti kapanpun untuk melakukan sebuah pekerjaan tersebut.

Penelitian ini akan mengimplementasikan metode pencarian *maximum-size matching* pada sebuah *bipartite graph* yang mengacu pada permasalahan penugasan bersyarat. Dalam penelitian ini dibahas algoritma *Hopcroft-Karp* untuk menyelesaikan permasalahan penugasan bersyarat tersebut dengan menggunakan bahasa pemrograman C++.

Dari serangkaian proses penelitian yang telah dilakukan, didapatkan kesimpulan bahwa algoritma yang dirancang sesuai dengan permasalahan ini dipengaruhi secara kuadratik baik oleh jumlah pekerjaan ataupun pekerjanya.

Kata Kunci—Algoritma *Hopcroft-Karp*, Grafik *Bipartite*, Masalah Penugasan Bersyarat, *Perfect Matching*, Teori Grafik

I. PENDAHULUAN

Dalam teori grafik, permasalahan penugasan bersyarat membahas bagaimana menemukan susunan pemberian tugas atau pekerjaan pada pekerja agar sebuah pekerjaan tersebut dapat dilakukan dengan seoptimal mungkin dengan beberapa konstrain permasalahan yang ada. Permasalahan penugasan bersyarat yang diangkat dalam penelitian ini diambil dari persoalan SPOJ Klasik 6819 yang berjudul *Yet Another Assignment Problem* dengan kode soal ASSIGN5. Permasalahan tersebut akan diselesaikan dengan pendekatan pencarian *Maximum-size Matching* pada sebuah *bipartite graph*.

Pada permasalahan *Yet Another Assignment Problem*, terdapat beberapa pekerjaan dan beberapa orang pekerja yang akan mengerjakan pekerjaan tersebut. Setiap pekerjaan harus dilakukan oleh semua orang yang ada serta setiap orang memiliki waktu yang dibutuhkan tersendiri dalam menyelesaikan pekerjaan tersebut. Setiap orang hanya dapat mengerjakan sebuah pekerjaan dan sebuah pekerjaan hanya dapat dikerjakan oleh satu orang dalam satu waktu. Pekerjaan yang dimaksud juga bersifat independen dalam artian dapat

dilakukan kapanpun oleh setiap orang. Semua orang juga dapat berhenti kapanpun untuk melakukan sebuah pekerjaan tersebut.

Diberikan m pekerjaan dan n pekerja, serta m baris kelompok bilangan. Setiap baris i berisi n buah bilangan dimana bilangan ke j -nya adalah A_{ij} yang berarti teman ke- j harus melakukan pekerjaan i selama A_{ij} menit. Diminta untuk mencari waktu minimal yang diperlukan untuk melakukan semua pekerjaan tersebut beserta susunan penugasan pada menit pertama yang mungkin terjadi.

II. TINJAUAN PUSTAKA

A. *Bipartite Graph*

Bipartite graph merupakan grafik spesial yang *vertex*-nya dapat dibagi menjadi dua himpunan *vertex* X dan Y dimana pada setiap himpunan *vertex*-nya, tidak ada *vertex* yang saling bertetangga. Dalam kasus khusus, *bipartite graph* dapat membentuk sebuah *complete bipartite graph*. *Complete bipartite graph* adalah sebuah *bipartite graph* dimana tiap *vertex* pada satu himpunan, bersinggungan dengan semua *vertex* pada himpunan yang lainnya.

B. *Matching*

Sebuah *matching* pada grafik $G = (V, E)$ adalah sebuah himpunan *edge* $M \subseteq E$ dimana tidak ada *edge* yang saling bersentuhan di dalam M . Untuk setiap *matching* M pada grafik $G = (V, E)$, kumpulan *edge* $e \in M$ dinamakan *matched edge*, sedangkan kumpulan *edge* $e \in E - M$ dinamakan *unmatched edge*. Sebuah *vertex* dikatakan *matched vertex* apabila *vertex* tersebut bersinggungan dengan *matched edge* pula. Sebaliknya *unmatched vertex* adalah *vertex* yang tidak bersinggungan dengan *matched edge*. Tiap *matched vertex* v memiliki sebuah *mate* atau pasangan yang terletak pada *endpoint* dari *matched edge* yang bersinggungan dengan v . Sebuah *matching* dapat dikatakan sebagai *perfect matching* apabila semua *edge* pada *matching* M dalam grafik $G = (V, E)$ mencakup setiap *vertex* dalam grafik atau tiap *vertex* dalam V bersinggungan dengan tepat satu *edge* pada M . Kardinalitas dari *matching* M adalah banyaknya *edge* pada M dan biasa dinotasikan dengan $|M|$. *Maximum-size matching* adalah sebuah *matching* M pada grafik $G = (V, E)$ yang kardinalitasnya tidak dapat dinaikkan lagi atau memiliki $|M|$ terbesar.

Bipartite matching merupakan kasus *matching* khusus yang melibatkan struktur *bipartite graph* didalamnya. *Bipartite graph* $G = (X \cup Y, E)$ dapat memiliki sebuah *complete matching* dengan kardinalitas $\min\{|X|, |Y|\}$.

C. Algoritma Hopcroft-Karp

Algoritma *Hopcroft-Karp* adalah algoritma yang memerlukan *bipartite graph* sebagai masukannya dan menghasilkan *maximum-size matching* didalam grafik tersebut sebagai keluarannya. Algoritma ini ditemukan John Hopcroft and Richard Karp pada tahun 1973.

Sebagian besar ide dasar dari algoritma ini hampir sama dengan teknik *inverting augmenting path* dalam mencari *maximum-size matching*, hanya saja algoritma ini mencari beberapa *augmenting path* sekaligus dalam sekali iterasi dengan menggunakan himpunan maksimal dari *augmenting path* terpendek.

Secara keseluruhan, algoritma ini berjalan pada sebuah grafik $G = (X \cup Y, E)$, dimana $|V|$ adalah jumlah *vertex* yang berada pada $X \cup Y$ dan $|E|$ adalah jumlah *edge* pada grafik, dengan kompleksitas $O(|V|\sqrt{|E|})$.

Dimisalkan terdapat sebuah grafik $G = (X \cup Y, E)$ dan M adalah *matching* yang di dapat dari grafik tersebut. Secara garis besar, algoritma ini berjalan sebagai berikut:

- Sebuah *breadth-first search* digunakan untuk membagi *vertex-vertex* menjadi beberapa *layer*. Sebuah *layer* dalam hal ini diartikan sebagai sebuah *level* dalam *tree*, dan setiap *level* dalam *tree* menunjukkan panjang *augmented path* yang dapat di bentuk. *Unmatched vertex* pada X digunakan sebagai *vertex* awal dari pencarian.
- Pada awal pencarian, hanya terdapat *unmatched edge* dikarenakan M masih merupakan himpunan kosong. Setelah itu, pada pencarian selanjutnya, melalui *unmatched vertex* pada X , pencarian path dilakukan dengan men-*reverse unmatched edge* dan *matched edge* secara bergantian. Pencarian akan berhenti pada sebuah *layer-k* dimana terdapat satu atau beberapa *unmatched vertex* pada Y tercapai.
- Setelah *layer* pada grafik tersebut terbentuk, melalui *depth-first search*, sebuah himpunan maksimal dari *augmenting path* terpendek yang memiliki panjang k dapat dicari dengan awal *unmatched vertex* pada X dan berakhir pada *unmatched vertex* Y .
- Untuk setiap *augmenting path* yang ditemukan, semua *unmatched vertex* pada tersebut diubah menjadi *matched vertex* begitu pula sebaliknya.

Algoritma ini berakhir apabila pada proses pencarian *breadth-first search* tidak menemukan *augmented path* yang dapat dibentuk.

III. STRATEGI PENYELESAIAN

A. Menentukan waktu minimum penyelesaian semua pekerjaan (α)

Sesuai pada deskripsi masalah, dapat diketahui bahwa setiap orang harus mengerjakan semua pekerjaan yang ada sesuai dengan waktu penyelesaiannya masing-masing dan setiap

2	2	4		
1	2	0		

(a)

2	2	4	0	0
1	2	0	0	5
5	0	0	3	0
0	4	0	4	0
0	0	4	1	3

(b)

Gambar 2. Proses *expanding-matrix* (a) *Matrix* awal (b) *Matrix* setelah mengalami proses *expanding-matrix*

B. Mencari maximum-size matching pada bipartite graph

Secara umum algoritma yang digunakan untuk menyelesaikan permasalahan ini berjalan sebagai berikut:

- Membentuk *bipartite* grafik $G = (X \cup Y, E)$ dengan X merupakan representasi dari kolom (pekerja) dan Y merupakan representasi dari baris (pekerjaan) dari *matrix* yang baru. Sebuah *edge* pada E menghubungkan *vertex* pada X dan Y apabila nilai $A_{i,j}$ pada *matrix* tersebut tidak 0.
- Mencari *maximum matching* yang terdapat pada grafik tersebut. Apabila sebuah kolom i memiliki pasangan dengan baris j , artinya pekerja ke- i mengerjakan pekerjaan ke- j dalam suatu waktu tersebut. Apabila $j > m$ maka pekerja tersebut tidak melakukan pekerjaan apapun di satuan waktu tertentu.
- Setiap baris j dan kolom i yang berpasangan, kurangi nilai $A_{i,j}$ pada *matrix* tersebut sebanyak 1.
- Ulangi setiap proses 1 – 3 hingga *matrix* tersebut menjadi *matrix* dengan semua isinya adalah 0.

Pada setiap proses yang terjadi, sebuah *perfect matching* terbentuk melalui grafik tersebut. Dikarenakan *matrix* adalah sebuah *matrix* persegi dengan ukuran $(m + n) * (n + m)$, maka semua baris memiliki pasangan pada kolom, begitu pula dengan kolom yang seluruhnya memiliki pasangan pada baris.

Seperti yang dijelaskan sebelumnya, diketahui bahwa jumlah dari semua elemen *matrix* tersebut adalah $(m + n) * \alpha$ dan setiap kali proses jumlah dari *matrix* tersebut berkurang sebanyak $m + n$, maka total proses yang diperlukan untuk menjadikan *matrix* tersebut menjadi *matrix* 0 adalah tepat sebanyak α .

IV. UJI COBA DAN ANALISIS

A. Uji Coba Kebenaran

Uji coba kebenaran dilakukan dengan mengirimkan kode sumber program kedalam situs penilaian *daring* SPOJ dan melakukan perbandingan hasil keluaran dari sistem dengan permasalahan. Permasalahan yang diselesaikan yaitu *Yet Another Assignment Problem* dengan kode soal ASSIGN5.



Gambar 3. Hasil Uji Coba pada Situs SPOJ

Kode sumber mendapat keluaran *Accepted* dari situs SPOJ. Hal ini berarti keluaran kode sumber telah sesuai dengan persoalan yang terdapat pada situs SPOJ. Waktu tercepat yang dibutuhkan program adalah 0.08 detik dan memori yang dibutuhkan adalah 64 MB.

Dilakukan pengujian sebanyak 20 kali pada situs penilaian *daring* SPOJ untuk melihat variasi waktu dan memori yang dibutuhkan program. Dari hasil uji coba sebanyak 20 kali, seluruh kode sumber program mendapat keluaran *Accepted* dengan waktu minimum sebesar 0.08 detik, waktu maksimum sebesar 0.12 detik dan waktu rata-rata sebesar 0.103 detik. Memori yang dibutuhkan program konstan sebesar 64 MB.

Tabel 1.
Hasil pengujian pada situs SPOJ sebanyak 20 kali

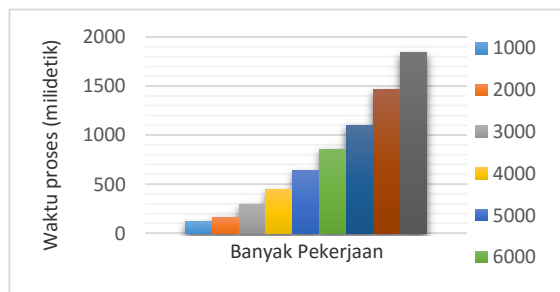
ID	RESULT	TIME	MEM
17014913	accepted	0.08	64M
17014912	accepted	0.10	64M
17014910	accepted	0.10	64M
17014909	accepted	0.09	64M
17014906	accepted	0.11	64M
17014903	accepted	0.11	64M
17014901	accepted	0.09	64M
17014899	accepted	0.10	64M
17014898	accepted	0.11	64M
17014896	accepted	0.12	64M
17014895	accepted	0.11	64M
17014893	accepted	0.09	64M
17014891	accepted	0.11	64M
17014890	accepted	0.10	64M
17014889	accepted	0.11	64M
17014886	accepted	0.12	64M
17014883	accepted	0.11	64M
17014882	accepted	0.10	64M
17014879	accepted	0.10	64M
17011446	accepted	0.10	64M

B. Uji Coba Efisiensi Waktu

Uji coba kinerja dilakukan dengan membuat data generator, untuk membangun contoh persoalan sesuai dengan batasan permasalahan. Kemudian dilakukan perbandingan kinerja program terhadap pertumbuhan jumlah pekerja dan pertumbuhan jumlah pekerjaan.

1) Pengaruh Banyaknya Pekerja Terhadap Waktu

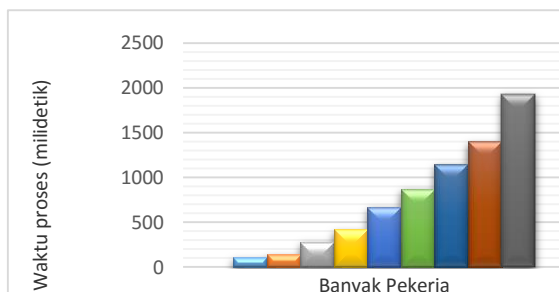
Pada uji coba ini banyaknya pekerjaan dibuat bervariasi antara 1000 hingga 9000. Jumlah pekerja ditetapkan sebanyak 1000. Waktu eksekusi program dicatat dalam satuan.



Gambar 4. Grafik Hasil Uji Coba Pengaruh Jumlah Pekerja Terhadap Waktu Eksekusi Program

Dapat dilihat bahwa grafik cenderung mendekati kurva kuadratik. Hal menunjukkan bahwa algoritma penyelesaian yang dipengaruhi secara kuadratik oleh jumlah pekerjaan. Pengaruh Banyaknya Pekerja Terhadap Waktu

Pada uji coba ini banyaknya pekerjaan dibuat bervariasi antara 1000 hingga 9000. Jumlah pekerja ditetapkan sebanyak 1000. Waktu eksekusi program dicatat dalam satuan.



Gambar 5. Grafik Hasil Uji Coba Pengaruh Jumlah Pekerja Terhadap Waktu Eksekusi Program

Dapat dilihat bahwa grafik cenderung mendekati kurva kuadratik. Hal menunjukkan bahwa algoritma penyelesaian yang dipengaruhi secara kuadratik oleh jumlah pekerja.

V. KESIMPULAN

Dari hasil uji coba yang telah dilakukan terhadap implementasi penyelesaian permasalahan *Yet Another Assignment Problem* dapat diambil kesimpulan sebagai berikut:

1. Permasalahan *Yet Another Assignment Problem* pada situs SPOJ dapat direpresentasikan sebagai sebuah *bipartite graph* serta Implementasi algoritma *Hopcroft-Karp* dapat menyelesaikan permasalahan tersebut dengan benar.
2. Kompleksitas waktu yang dibutuhkan untuk seluruh proses sistem adalah $O((m + n)^2)$ pada m buah pekerjaan dan n orang pekerja sehingga algoritma ini dipengaruhi secara kuadratik baik oleh jumlah pekerja maupun jumlah pekerjaan.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas segala karunia dan rahmat-Nya yang telah diberikan selama ini. Penulis juga mengucapkan terimakasih kepada orang tua, saudara, keluarga, Bapak Arya Yudhi Wijaya dan Bapak Rully Soelaiman selaku pembimbing penulis dan semua pihak yang tidak bisa penulis sebutkan satu persatu yang telah memberikan semangat serta motivasi sehingga penulis dapat menyelesaikan penelitian ini.

DAFTAR PUSTAKA

- [1] SPOJ, "The dilemma of Idli," March 2014. [Online]. Available: <http://www.spoj.com/problems/WPC5G/>.
- [2] S. Halim and F. Halim, "Competitive Programming 3," Singapore, Lulu Publisher, 2013.
- [3] D. Jungnickel, *Graphs, Networks and Algorithms*, 4th Edition ed., Berlin: Springer, 2012.
- [4] SPOJ, "*Yet Another Assignment Problem*," June 2010. [Online]. Available: <http://www.spoj.com/problems/ASSIGN5/>.
- [5] J. Bondy and U. Murty, *Graph Theory*, 3rd ed., Berlin: Springer, 2008.
- [6] H. A. Taha, *Operations Research: An Introduction*, 8th Edition ed., Upper Saddle River: Prentice Hall, 2007.