Aplikasi Sistem Penjadwalan Praktikum dengan Metode *Bipartite Graphs*

Studi Kasus : Laboratorium Terpadu Teknik Informatika UII

A'mal Sholihan
Jurusan Teknik Informatika, FTI
Universitas Islam Indonesia
Jl.Kaliurang KM 14 Yogyakarta
Telp. (0274) 898444
amalsholihan@gmail.com

Nielsa Maulida
Jurusan Teknik Informatika, FTI
Universitas Islam Indonesia
Jl.Kaliurang KM 14 Yogyakarta
Telp. (0274) 898444
nielsa.maulida@gmail.com

Hendika Andra Saputra
Jurusan Teknik Informatika, FTI
Universitas Islam Indonesia
Jl.Kaliurang KM 14 Yogyakarta
Telp. (0274) 898444
hendikaandra@yahoo.com

Feri Wijayanto
Jurusan Teknik Informatika, FTI
Universitas Islam Indonesia
Jl.Kaliurang KM 14 Yogyakarta
Telp. (0274) 898444
feri.wijayanto@uii.ac.id

***Abstrak*—Penjadwalan merupakan permasalahan yang selalu dihadapi sebelum kegiatan belajar-mengajar dimulai. Untuk memperoleh jadwal yang optimal, setiap kelas diampu oleh sejumlah asisten sesuai kebutuhannya dan setiap asisten mengajar sesuai kuota mengjar yang dimiliki masing-masing asisten. Di penelitian kali ini kami mengambil metode *bipartite graph* untuk memilih asisten dan kelas yang sesuai. Metode ini mengubah permasalah kali ini menjadi sebuah *graph* yang dapat dibagi dua subset. Subset pertama, mempunyai *node* kelas yang tersedia. Dan subset kedua, mempunyai *node* kelompok asisten dalam setiap kelas. Dengan menerapkan metode ini, kedua subset akan saling dicocokkan. Sehingga permasalahan penjadawalan dapat diselesaikan ketika *bipartite graph* memenuhi kondisi *complete matching*.**

Kata kunci— Bipartite graph, Complete Matching, Scheduling

I. PENDAHULUAN

Baker (1974) mengatakan bahwa penjadwalan merupakan alokasi dari sumber daya terhadap waktu untuk menghasilkan sebuah kumpulan pekerjaan [1]. Penjadwalan juga didefinisikan sebagai rencana pengaturan urutan kerja serta pengalokasian sumber, baik waktu maupun fasilitas untuk setiap operasi yang harus diselesaikan [2]. Penjadwalan dapat didefinisikan sebagai sebuah permasalahan yang memiliki empat parameter: himpuan berhingga waktu (T), himpunan berhingga sumber daya (R), himpunan berhingga pertemuan (M), dan himpunan berhingga batasan (C) [3]. Sehingga, dapat dikatakan bahwa penjadwalan merupakan pengalokasian segala sumber daya, sesuai dengan waktu dan batasan yang harus dipenuhi.

Penjadwalan merupakan salah satu permasalahan yang harus dihadapi sebelum kegiatan belajar mengajar dimulai dan harus dipastikan optimal. Dalam kasus ini, penjadwalan disebut optimal jika setiap kelas diampu oleh sejumlah asisten sesuai kebutuhannya dan setiap asisten mengajar sesuai kuota mengajar yang dimiliki masing-masing asisten dengan seluruh batasan, kendala dan parameter yang kadang membuat penjadwalan berbenturan satu sama lain. Tanpa adanya penjadwalan yang optimal, maka sumber daya manusia, maupun kelas tidak akan optimal. Bahkan, dapat mengganggu kegiatan belajar mengajar. Maka dari itu, penjadwalan yang optimal harus dimiliki sebelum kegiatan belajar mengajar dilaksanakan. Akan tetapi, memperoleh penjadwalan yang optimal bukanlah suatu hal yang mudah. FTI UII merupakan salah satu universitas yang tak lepas dari permasalahan ini. Karena kegiatan praktikum di FTI UII Jurusan Informatika diampu oleh lima laboratorium yaitu Sirkel, PIT, KSC, Jarkom dan GMM yang sering disebut dengan laboratorium terpadu Jurusan Teknik Informatika. Setiap praktikum yang dilaksanakan harus sesuai dengan asisten yang mengajar dan waktu yang dimiliki oleh asisten tanpa menggangu jadwal kuliah yang dimilikinya.

Permasalahan penjadwalan yang sebenarnya sangat kompleks, bahkan kadangkala batasannya seringkali berubah-ubah. Sehingga, sistem yang dibangun juga harus dapat berubah-ubah atau dinamis. Penjadwalan kuliah termasuk sebuah permasalahan NP-*hard*, sulit diselesaikan dengan menggunakan metode yang konvensional dan waktu komputasi yang dibutuhkan untuk mendapatkan solusi yang optimal meningkat eksponensial seiring besarnya permasalahan [3]. Hal ini sama saja dengan penjadwalan praktikum, permasalahan ini sulit diselesaikan dengan metode konvensioanal. Bahkan, Penjadwalan yang dibuat oleh asisten kadangkala tidak

optimal dan harus terus-menerus berubah-ubah untuk mencapai optimal. Penjadwalan untuk lab-informatika terpadu UII, terkadang lebih kompleks dan sering berubah-ubah. Karena setiap kelas dapat diampu lebih dari satu asisten dan penjadwalan yang dibuat harus mampu mengalokasikan waktu yang dimiliki asisten sepenuhnya untuk mengajar dan kuliah yang dijalaninya.

Permasalan ini, bisa diselesaikan dengan cara penyesuaian asisten dengan kelas yang tepat. Sehingga dapat diperoleh suatu jadwal yang tepat bagi asisten atau praktikum itu sendiri. Bermacam-macam metode telah dikemukakan untuk menyelesaikan metode ini, seperti *bipartite graphs* [4], *simulated annealing*, tabu *search* dan algoritma genetika [5]

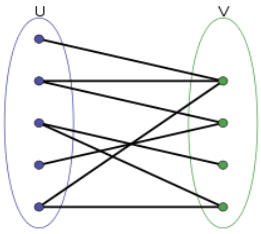
II. KAJIAN TEORI

A. *Bipartite Graph*

Bipartite graph dapat disebut juga sebagai *bigraph*. *Bigraph* ini merupakan satu set simpul *graph* yang dapat didekomposisi menjadi dua *disjoint* subset yang masing-masing subset tersebut *disjoint*. Sehingga, dapat dikatakan bahwa *bipartite graph* tidak dapat mempunyai siklus.

Bipartite graph equivalen dengan dua *graph* warna dimana semua siklus dari *bipartite graph* yang mempunyai panjang [6]. Kedua set yang dimiliki oleh *bipartite graph* dapat juga disebut sebagai suatu *coloring graph* dengan dua warna [7]. Dapat dimisalkan, jika satu node mempunyai warna biru dan node yang lain berwarna hijau. Dan keduanya saling dihubungkan oleh *Edge*.

Sebaliknya dalam suatu kasus *graph non-bipartite*, tidak dapat dilakukan pemberian 2 warna. Contohnya dalam suatu segitiga, jika salah satu *node* berwarna biru dan *node* lain berwarna hijau. Maka *node* ketiga, tidak dapat berwarna biru maupun hijau. Karena *node* ketiga saling terhubung dengan *node* biru dan *node* hijau. Jika kedua subset dari *bipartite* mempunyai kardinalitas yang sama maka dapat disebut dengan *balanced bipartite graph*. Dan jika kedua sisi subset mempunyai drajat yang sama maka dapat disebut *biregular graph*.



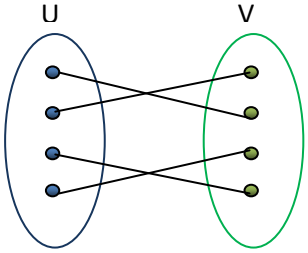
Gambar 1. Contoh *Bipartite Graph* Tanpa *Cycles*

Contoh diatas merupakan gambar dari *bipartite graph* yang mempunyai 2 subset *U* dan subset V. *Graph* seperti ini sering ditulis G=(*U, V, E*), *graph* yang mempunyai subset *U, V* dan dihubungkan oleh *Edge E*.

B. *Complete Matching*

Dalam suatu *graph matching* atau pemasangan antara satu *node* dengan *node* yang lain merupakan suatu hal yang tidak bisa dilepaskan. Berbagai algoritma untuk penyelesaian permasalahan pemasangan ini telah banyak dikemukakan. Seringkali timbul suatu permasalahan, seperti *maximum mathcing* (*mathcing* dengan menggunakan *edge* sebanyak-banyaknya), *maximum weight mathcing*, dan *stabble marriage* [8].

Dalam peneltian kali ini, penjadwalan akan optimal jika *bipartite graph* memenuhi kondisi *Complete mathcing*. Kondisi ini dapat dicapai ketika setiap *node* di salah satu subset mempunyai satu pasangan atau berpasangan *one to one* dengan *node* dari subset lainnya [9]. Berikut adalah contoh *bipartite graph Complete matching*.



Gambar 2. Contoh *Bipartite Graph* dengan *Complete Mathcing*

Selain *Complete mathcing*, juga terdapat kondisi *Incomplete Matching*. kebalikan dari *matching complete*, dimana ada satu atau beberapa *node* dalam salah satu subset tidak memiliki pasangan dengan *node* dari subset lainnya.

III. DESAIN SISTEM

Sistem ini dikembangkan untuk membantu dalam pembuatan jadwal mengajar asistan laboraturium informatika terpadu. Hal ini dikarenakan sering terjadi kesalahan jadwal, banyaknya kelas dan praktikum yang diampu, dan terbatasnya waktu yang dimiliki asistan dalam melaksanakan kegiatannya. Jadwal yang dirancang oleh sistem ini dengan mempertimbangkan berbagai aturan dalam penjadwalan yang sebelumnya telah dibuat oleh laboraturium informatika terpadu. Berikut ini adalah kriteria pencocokan yang digunakan di kelas praktikum lab informatika terpadu:

a) Jadwal Kuliah Asisten

Sistem ini akan membuat jadwal yang disesuaikan dengan jumlah mata kuliah yang diambil oleh asisten disetiap semester. Hal ini menyebabkan waktu yang digunakan asisten dalam pembuatan jadwal semakin sedikit.

b) Jumlah Mengajar Setiap Asisten

Setiap asisten mempunyai jumlah kouta mengajar yang berbeda-beda, misalnya asisten yang senior akan

mendapatkan jumlah mengajar yang lebih sedikit daripada asisten yang baru.

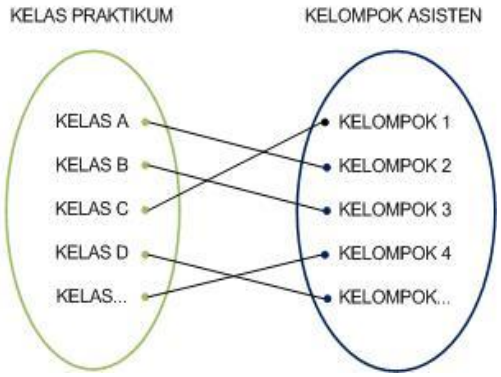
c) Jumlah Praktikan

Jumlah praktikan setiap asisten juga dipertimbangkan dalam proses perancangan sistem ini. Jika satu kelas praktikum terisi penuh maka asisten yang diperlukan akan lebih banyak dibandingkan dengan kelas yang terisi sebagian oleh praktikan. Hal ini dilakukan agar kegiatan belajar dan mengajar dapat dilakukan dengan efisien dan efektif.

d) Jumlah Praktikum

Dalam pembuatan jadwal, jumlah praktikum yang tersedia juga dipertimbangkan. Karena dalam setiap semesternya jumlah praktikum pada masing-masing laboraturium itu berbeda-beda. Misalnya pada semester genap, laboraturium SIRKEL mengampu dua praktikum sekaligus, sedangkan PIT hanya mengampu satu praktikum.

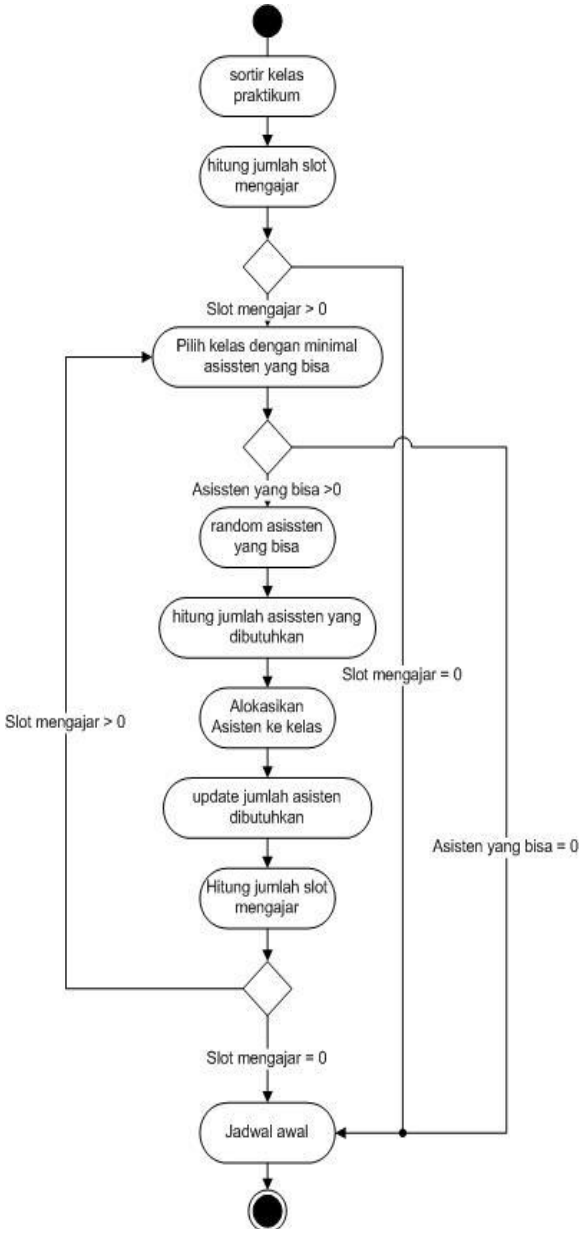
Berikut merupakan *bipartite graph* yang harus dipenuhi sistem untuk mencapai *complete matching* agar jadwal dapat optimal.



Gambar 3. *Bipartite Graph* untuk Penjadwalan Praktikum

Gambar 3 menjelaskan bahwa subset pertama atau subset kelas praktikum dicocokkan dengan satu subset di subset kelompok asisten. Langkah ini ditempuh agar setiap kelompok asisten yang memenuhi kriteria diatas dapat dicocokkan dengan kelas yang sesuai.

Untuk membuat kelompok asisten dan kelas diatas, sistem menggunakan Diagram Aktivitas pada Gambar 4.

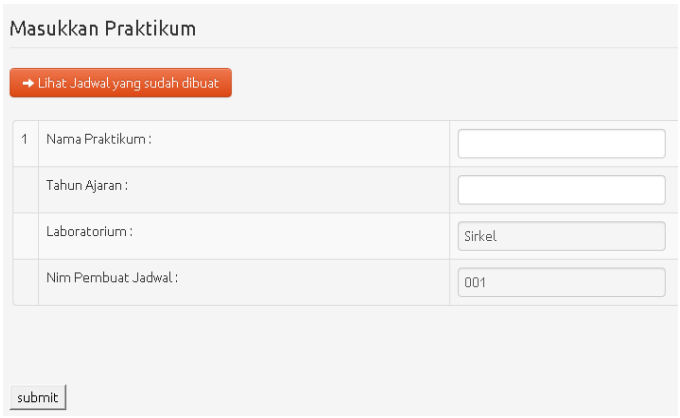


Gambar 4. Diagram Aktivitas Pengalokasian Kelas Berdasarkan Asisten yang Bisa

Gambar 4 menjelaskan urutan aktivitas yang dilakukan untuk membuat jadwal awal. Dimana jadwal awal ini, akan dibandingkan dengan jadwal lain jika kondisi *complete matching* belum terpenuhi sebelumnya.

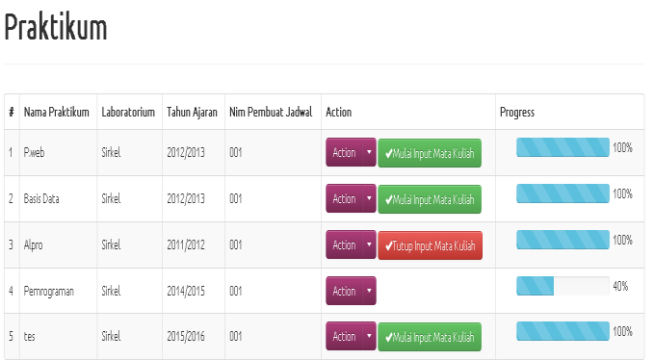
IV. IMPLEMENTASI

Berikut adalah hasil implementasi dari sistem yang telah dibuat. Sistem ini mempunyai sistem untuk menambah praktikum, menambah kelas, menambah asisten, dll.



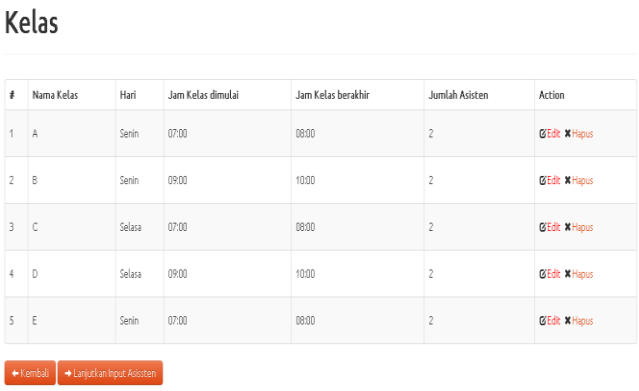
Gambar 5. *Form* untuk Membuat Praktikum Baru

Gambar 5 menunjukkan halaman *form* untuk membuat praktikum baru. Dimana *form* tersebut terdiri dari nama praktikum, tahunajaran laboratorium dan nim pembuat. Halaman ini muncul ketika user mempunyai hak akses untuk membuat jadwal.



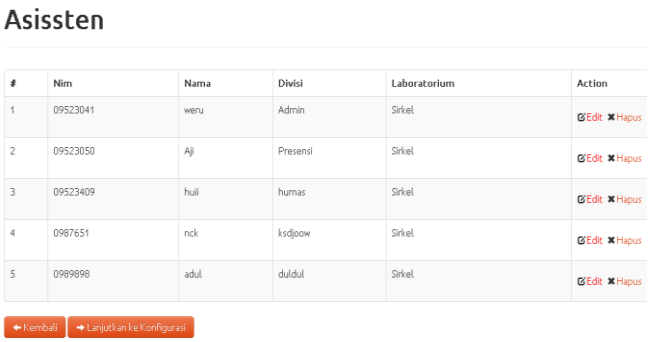
Gambar 6. Daftar Praktikum yang Telah Dibuat

Ketika praktikum telah dibuat maka, daftar diatas akan bertambah. Akan tetapi masih menunjukkan progress 20% dan *link* untuk memulai *input* mata kuliah pun belum ada. Sebaliknya ketika *input* kelas dan *input* asisten sudah dilakukan maka progress akan bertambah dan bisa memulai *input* mata kuliah oleh asisten.



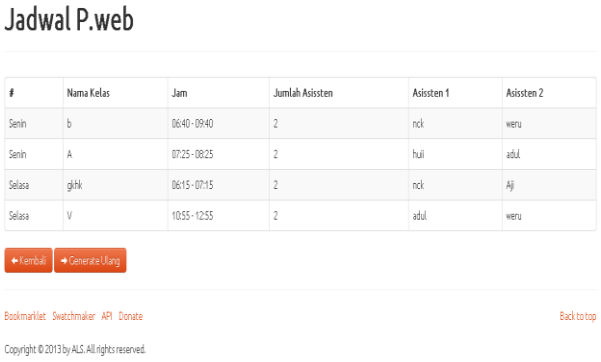
Gambar 7. Daftar Kelas Praktikum

Gambar di atas menunjukkan informasi mengenai kelas apa saja yang telah dimasukkan. Informasi yang ditampilakan meliputi nama kelas, hari, jam dan jumlah asisten. Di sini asisten pembuat jadwal juga dapat menghapus atau mengubah daftar kelas yang telah dibuat sebelumnya. Begitu juga dengan daftar asisten. Menunjukkan informasi tetang asisten yang telah dimasukkan ke sistem.



Gambar 8. Daftar Asisten Praktikum

Setelah semua kelas dan asisten praktikum telah dimasukkan ke sistem maka asisten dapat mengatur konfigurasi jadwal yang akan dibuat seperti mengatur jumlah mengajar masing-masing asisten pengampu praktikum. Saat konfigurasi sistem akan menawarkan jumlah mengajar setiap asisten diatur oleh sistem atau diatur sendiri. Karena terkadang dalam suatu praktikum setiap asisten mempunyai kesempatan mengjar berbeda-beda.



Gambar 9. Jadwal Akhir Praktikum

Setelah semua *input* telah dimasukkan maka asisten diperbolehkan untuk melakukan *generate* jadwal. Gambar diatas merupakan hasil dari *generate* jadwal praktikum P.web

V. PENGUJIAN

Dalam penelitian kali ini kami menggunakan metode *bipartite graphs*. Dengan metode ini, permasalahan penjadwalan dapat diselesaikan dengan membaginya menjadi dua subset utama dimana dua subset ini harus mempunyai anggota. Kedua subset ini akan saling dihubungkan dan disesuaikan dengan *edge*. Tetapi, kedua subset ini juga harus memenuhi kondisi yang telah ditentukan sebelumnya. Sehingga hasil akhir dari pesuaian kedua subset tersebut dapat memenuhi kondisi *complete mathcing*. Kondisi *complete matching* dapat dipenuhi ketika setiap anggota di subset pertama memupunyai satu pasangan dengan salah satu anggota subset kedua.

Dalam implementasi, subset pertama kita sebut subset *U* dan subset kedua kita sebut subset *V*. Sehingga dapat kita tuliskan G=(*U, V, E*). Dimana subset *U* dan subset *V* dipasangan dengan *Edge E*. Subset *U* mempunyai anggota setiap kelas dalam suatu praktikum. Akan tetapi sebelum setiap kelas dimasukkan ke subset *U*, tentukan terlebih dahulu manakah kelas yang saling bertabrakan. Karena jika terdapat kelas yang bertabrakan, maka asisten kedua kelas ini tidak boleh sama. Misalkan, jika kelas A dimulai jam 07:00 dan diakhiri jam 08:00 sedangkan kelas B dimulai jam 07:30. Hal ini menyebabkan asisten yang mengajar di kelas A tidak bisa mengajar di kelas B.

Sedangkan subset *V* atau subset kedua mempunyai anggota asisten atau group asisten jika terdapat lebih dari satu asisten dalam kelas praktikum. Group asisten ini akan disusun atau dipilih oleh sistem seperti ketentuan atau kondisi yang harus dipenuhi. Kondisi tersebut ialah :

- Jumlah asisten yang berada di *group* asisten harus sama dengan jumlah asisten yang di butuhkan oleh kelas praktikum.
- Asisten tidak bisa mengajar di dua atau lebih kelas yang mempunyai waktu yang sama atau saling bertabrakan.
- Dalam satu *group* asisten tidak boleh terdapat asisten yang sama.
- Setiap asisten mempunyai jumlah mengajar yang telah ditentukan sebelumnya

- Jadwal mengajar asisten praktikum, tidak boleh bersamaan dengan jadwal kuliah asisten

Setelah semua kondisi tersebut dapat terpenuhi maka sistem dapat memulai *generate* asisten untuk menjadi anggota pada subset *V*. Maka sistem akan mulai menyesuaikan antara subset *U* dan subset V dengan *Edge E* agar dapat menjadi *graph bipartite* yang memenuhi *complete mathcing*. Urutan aktivitas ini dapat dilihat pada Gambar 4.

Uji coba dilakukan pada komputer dengan perangkat keras : Intel Pentium Centrino T6500 dengan ram 2GB. Sedangkan perangkat lunak komputer ini menggunakan Windows 7 dengan *browser* Google Chrome dan basis data MYSQL.

Hasil lima kali eksekusi penjadwalan dengan lima kelas dan disetiap kelas mempunyai dua asisten. Sedangkan terdapat empat asisten yang akan mengajar, menunjukkan bahwa sistem dapat mengalokasikan bahwa setiap asisten dapat mengajar dua kali atau tiga kali mengajar. Dan setiap kelas dapat teralokasi dengan baik. Berikut adalah tabel hasil eksekusi.

TABEL I. HASIL EKSEKUSI PENYUSUNAN JADWAL

No.	Jumlah Asisten	Jumlah Kelas	Slot kelas kosong	Waktu (*ms*)
1.	4	5	0	895
2.	4	5	1	492
3.	4	5	1	606
4.	4	5	1	678
5.	4	5	0	566

Dari tabel diatas dapat dilihat bahwa, dengan eksekusi sebanyak lima kali rata-rata waktu yang dibutuhkan untuk membuat jadwal awal 647,4 ms. Sedangkan, untuk mencapai jadwal yang optimal sistem hanya memerlukan waktu 1319 ms. Waktu ini diperoleh dari 2 kali eksekusi penjadwalan yang telah mencapai jadwal yang optimal.

TABEL II. HASIL EKSEKUSI OPTIMASI JADWAL

No	Jumlah Asisten	Jumlah Kelas	Slot kelas kosong	Waktu (*ms*)
1.	4	5	1	686
2.	4	5	0	633

VI. SIMPULAN

Jadwal yang didapatkan dari metode bipartite graph menunjukkan bahwa penjadwalan dapat mencapai optimal tanpa memerlukkan waktu yang lama. Waktu yang dibutuhkan untuk membuat jadwal rata-rata 647,4 ms. Sedangkan untuk mencapai jadwal yang optimal hanya dibutuhkan 2 kali perulangan dengan total waktu 1319 ms.

Hasil ini memperlihatkan bahwa penjadwalan dengan metode *bipartite graph* cukup baik. Tanpa membutuhkan waktu yang lama dan proses yang panjang, jadwal dapat mencapai titik optimal.

DAFTAR PUSTAKA

[1] Baker, Kenneth R., *Introduction To Sequencing And Scheduli*ng,. (1974).

[2] T.E.Vollman,W.L. Berry, And D.C. Whybark, *Manufacturing Planning And Control System*, 4th Edition (Burr Ridge, Ll:Lrwin/Mcgraw-Hill, 1997).

[3] Rochman, Abdul, *Penjadwalan Kuliah Menggunakan Metode Constraints Programming dan Simulated Annealing*. Jurusan Teknik Informatika, Fakultas Teknologi Industri,Universitas Trisakti (2012).

[4] Hung Q, Ngo.(2004) ”Matchings on bipartite graphs”http://www.cse.buffalo.edu/~hungngo/classes/2004/594/notes/matching.pdf (diakses tanggal 2 Januari 2013).

[5] Al-Milli, N. R., “Hybrid Genetic Algorithms with Great Deluge For Course Timetabling”, IJCSNS International Journal of Computer Science and Network Security, 10(4): 283-288 (2010).

[6] Skiena Steven S, “Implementing discrete mathematics: combinatorics and graph theory with Mathematica”, Addison-Wesley (1990).

[7] Weisstein, *Eric*."Bipartite Graph” http://mathworld. wolfram.com/BipartiteGraph.html (diakses tanggal 5 Januari 2013).

[8] Ahuja, Ravindra K.; Magnanti, Thomas L.; Orlin, James B,"12. Assignments and Matchings", Network Flows: Theory, Algorithms, and Applications, Prentice Hall, pp. 461–509, (1993).

[9] HEGARTYMATHS. “Bipartite Graphs/Matching (Intro)-Tutorial 12 D1 Edexcel” http://www.youtube.com/watch?v=JpapV5DrBek (diakses tanggal 4 Januari 2013).