

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/319122545>

PENJADWALAN KULIAH OTOMATIS DENGAN CONSTRAINT PROGRAMMING

Article · January 2008

CITATIONS

4

READS

1,410

1 author:



Yumarsono Muhyi

9 PUBLICATIONS 5 CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:



Skripsi Sistem Komputer [View project](#)

PENJADWALAN KULIAH OTOMATIS DENGAN CONSTRAINT PROGRAMMING

Yumarsono Muhyi,¹⁾

1) STMIK Supra, Jakarta - Indonesia, email: yumarsono.muhyi@supra.ac.id, y.muhyi@gmail.com

Abstract: Scheduling is common at work, which is to arrange resources and tasks in order to make them placed appropriately. The general problem is the limited resources, while the tasks to be accomplished are abundant, and it makes the scheduling harder to solve and sometimes trivial too. There are techniques and approaches to solve scheduling problem, one of them is Constraint Programming. By using Constraint Programming, scheduling with resources and tasks with their own prerequisites and limitations, can be modelled with ease and solved automatically, instantly and repeatedly. This article will focus on lecturing scheduling problem, with its own specific characteristics and approaches, covering the modelling aspects and the implementation aspects, without elaborating the performance aspects.

Keywords: Schedule, Problem, Lecture, Constraint, Solution, Automation, Academic

Masalah yang umum terjadi hampir setiap waktu tentang penjadwalan kuliah, adalah bahwa sumber-sumber daya terbatas sementara perkuliahan yang harus dilayani berjumlah banyak. Contoh-contoh keterbatasan sumber daya tersebut adalah:

- jumlah kelas yang terbatas,
- jumlah dosen yang terbatas,
- dosen bisa atau bersedia mengajar di waktu-waktu yang tertentu,
- dan batasan-batasan lainnya, yang akan dibahas secara mendetail dalam artikel ini.

Ada beberapa teknik dan pendekatan dalam menyelesaikan masalah penjadwalan, yang masing-masing memiliki keunggulan. Salah satu teknik itu adalah *Constraint Programming*, dan teknik inilah yang akan diterapkan dalam artikel ini.

Constraint Programming adalah sebuah pendekatan perhitungan atau komputasi matematis atas masalah-masalah yang berkaitan dengan batasan-batasan dari variabel-variabel, dengan tujuan mencari solusi yang memenuhi syarat-syarat tersebut.

Masalah yang harus diselesaikan dalam *Constraint Programming* disebut sebagai *Constraint Satisfaction Problem* (CSP) (Barták, 1999) atau *Constraint Logic Programming* (CLP) (Abdennadher, 2001). Untuk selanjutnya, penulis akan menggunakan istilah *Constraint Programming* atau singkatan CSP.

Pemodelan CSP yang lengkap terdiri dari:

1. Variable, atau variabel yang umumnya berjumlah lebih dari satu.
2. Domain, atau semua kemungkinan nilai untuk tiap variabel yang ada.
3. Constraints, atau batasan-batasan yang harus dipenuhi untuk tiap variabel yang ada.

Yang akan menjadi solusi (atau set solusi, atau kumpulan solusi) dari CSP, adalah nilai-nilai dari variabel-variabel itu sendiri. Solusi yang didapat tersebut ada beberapa kemungkinan:

1. Satu set solusi saja, biasanya dari yang pertama didapat dari pencarian.
2. Seluruh set solusi yang bisa diperoleh.

3. Set solusi yang paling optimum, yang bisa berarti maksimum, minimum, atau mendekati suatu nilai tertentu.

Constraint Programming ini beririsan dengan *Operational Research* (Riset Operasi) yang kental dengan solusi *Artificial Intelligence* (Kecerdasan Buatan) (Barták, 1998). Namun, *Constraint Programming* lebih baik dalam hal pemodelan, karena tidak terkait dengan “bagaimana” atau “cara” mendapatkan solusi.

Dengan *Constraint Programming*, masalah didekati dan dicari solusinya cukup hanya dengan menspesifikasikan batasan-batasan yang harus diselesaikan. *Constraint Programming* tidak butuh pemodelan yang detail dari suatu masalah. Karena itulah, *Constraint Programming* lebih dapat diterima dan lebih mudah diterapkan dalam lingkungan kerja dan industri.

Tujuan akhir dari artikel ini adalah untuk memberikan suatu rekomendasi pemodelan sistem dan solusi atas masalah penjadwalan kuliah, yang selalu berulang terjadi setiap semester. Dan dari rekomendasi ini juga, diharapkan metode dan pemodelan ini bisa diterapkan ke dalam masalah-masalah pemodelan-pemodelan sistem lain yang serupa dengan penjadwalan kuliah.

METODE PENELITIAN

Penelitian ini dilakukan dengan dua metode: studi literatur dan observasi. Studi literatur difokuskan ke masalah-masalah riset operasi, khususnya pada bidang *Constraint Programming*.

Observasi dilakukan dengan mengamati faktor-faktor apa saja yang berpengaruh dalam masalah penjadwalan perkuliahan. Observasi terutama dilakukan di institusi tempat penulis bekerja, yaitu STMIK Supra, yang penulis anggap telah mewakili keumuman kondisi tentang penjadwalan kuliah.

Dari penelaahan atas berbagai literatur dan observasi di lapangan, maka penulis menganalisa dan melakukan pemodelan atas penjadwalan perkuliahan.

Dari pemodelan itu, penulis juga melakukan uji coba model dengan menggunakan bahasa pemrograman Java, dan dengan menggunakan komponen alat bantu untuk penghitungan *Constraint Programming* bernama Choco (Choco, 2007).

Komponen-komponen Java yang dapat digunakan untuk perhitungan numerik dan penyelesaian CSP ada beberapa. Selain Choco, ada komponen lain yang telah penulis coba, yaitu JACK (Abdennadher, 2002).

Penulis memilih Choco karena lebih mudah dalam penggunaannya, sehingga penulisan CSP ke dalam baris-baris kode Java menjadi lebih mudah, intuitif, dan cepat. Dengan demikian, penulis bisa fokus ke dalam masalah pemodelan CSP dari penjadwalan kuliah.

Analisa dan Pemodelan

Pemodelan variabel yang digunakan untuk solusi masalah penjadwalan kuliah pada artikel ini akan dipaparkan pada bagian akhir dari bagian ini. Variabel akan dapat dimodelkan dengan sempurna, setelah data-data (atau dimensi) yang terlibat dan digunakan dalam masalah penjadwalan dipaparkan dengan baik.

Adapun data-data dan syarat-syarat (atau *constraints*) yang harus dipenuhi dalam CSP untuk masalah penjadwalan kuliah ini, secara kualitatif adalah sebagai berikut:

1. Tiap guru (atau dosen) mendapatkan tugas mengajar materi-materi tertentu.
2. Tiap kelas adalah untuk tingkat yang tertentu, dan diadakan pada shift tertentu. Kelas di sini bisa berarti “ruangan” atau “ruang kelas” secara fisik, dan bisa juga dimaknai sebagai “kelompok siswa”.
3. Tiap materi (atau subjek pelajaran) ditujukan untuk tingkat tertentu.
4. Tiap guru memiliki kesanggupan mengajar yang tertentu, baik itu shift, hari, maupun jamnya. Dan pada umumnya, syarat ini yang paling menyulitkan dalam urusan penjadwalan kuliah.

Selanjutnya, akan dipaparkan pemodelan-pemodelan matematis dari permasalahan di atas, agar menjadi CSP yang sempurna dan dapat diselesaikan.

Dimensi

Pemodelan penjadwalan kuliah dimulai dengan penelaahan data-data (atau dimensi-dimensi) yang umum dan mendasar. Arti dari dimensi di sini adalah himpunan objek yang sama, dan terlibat dalam penjadwalan kuliah.

Dalam perhitungan, dimensi-dimensi ini menggunakan indeks angka integer, yang merepresentasikan isi anggotanya. Jadi, dimensi ini lebih tepat jika dimodelkan sebagai *array*.

Misalkan: dimensi D adalah himpunan semua huruf kapital, maka $D = \{A, B, C, D, \dots, Z\}$.

Dalam perhitungan *Constraint Programming*, maka D akan menjadi *array*, dan dinotasikan sebagai:

- $D[0] = A$
- $D[1] = B$
- $D[2] = C$
- $D[3] = D$
- ...
- $D[23] = Z$.

Cara notasi dimensi-dimensi dasar ke dalam *array*, persis sama seperti dalam penjelasan di atas. Dimensi-dimensi dasar dalam penjadwalan kuliah di sini adalah:

1. Shift (S), adalah shift-shift yang akan dibuka dalam kalender akademik, misalnya:
 - Shift Pagi
 - Shift Siang
 - Shift Malam
 - Dan sebagainya.
2. Kelas (K), adalah jumlah total kelas yang akan dibuka, misalkan:
 - Kelas 1.1
 - Kelas 2.3
 - Dan sebagainya.
3. Jam (J), adalah jam-jam kuliah yang akan diadakan untuk tiap shift, dan 1 jam di sini artinya bernilai 1 kredit, misal:
 - Jam 8:00-9:00
 - Jam 9:00-10:00,
 - Dan seterusnya.
4. Guru (G), adalah guru-guru atau dosen-dosen yang akan mengajar, misal:
 - Ahmad
 - Joni
 - Badru
 - Dan sebagainya.
5. Materi (M), adalah materi-materi atau subjek-subjek perkuliahan yang ada, misal:
 - Web Programming
 - Decision Support
 - Dan selainnya.
6. Tingkat (T), adalah tingkatan-tingkatan atau jenjang-jenjang yang ada, misal:
 - Tingkat 1
 - Tingkat 2
 - Dan seterusnya.
7. Hari (H), adalah hari-hari perkuliahan yang ada, misal:
 - Senin
 - Selasa
 - Rabu
 - Kamis
 - Jumat
 - Dan seterusnya.

Hubungan Antar Dimensi

Dari masing-masing dimensi-dimensi dasar di atas, ada hubungan-hubungan khusus dari beberapa

dimensi yang harus berkaitan. Hubungan-hubungan antar dimensi tersebut adalah:

1. Guru-Materi (GM), adalah penugasan guru ke materi-materi yang diajarkannya, misalkan:
 - Ahmad mengajar Web Programming dan Decision Support
 - Joni mengajar Decision Support dan Operating System
 - dan seterusnya
2. Kelas-Shift (KS), adalah penetapan shift perkuliahan dari kelas, misal:
 - Kelas 1.1 pada Shift Malam
 - Kelas 2.3 pada Shift Pagi
 - Dan seterusnya.
3. Kelas-Tingkat (KT), adalah penetapan kelas dengan tingkatnya, misal:
 - Kelas 1.1 adalah Tingkat 1
 - Kelas 2.3 adalah Tingkat 2
 - Dan seterusnya
4. Materi-Tingkat (MT), adalah penetapan materi dengan tingkatnya, misal:
 - Web Programming adalah untuk Tingkat 1
 - Decision Support adalah untuk Tingkat 2
 - Dan seterusnya.
5. Guru-Shift -Hari-Jam (GSHJ), adalah data waktu mengajar tiap guru, sampai ke jam setiap harinya.
 - Ahmad sanggup mengajar pada setiap hari kerja, dari jam 8:00 sampai jam 11:00
 - Badru hanya bisa mengajar hari Rabu saja, mulai jam 8:00 sampai jam 17:00
 - Joni hanya ingin mengajar pada hari Senin sepanjang hari, Selasa di bawah jam 12:00, dan lainnya antara jam 17:00 sampai jam 20:00
 - Dan seterusnya.

Cara notasi hubungan antar dimensi ini ke dalam bentuk *array*, menjadi *array* berdimensi sejumlah hubungan dimensi yang terlibat. Adapun nilai anggota dari *array* ini, bernilai angka 1 (satu) atau 0 (nol). Satu, jika ada relasi antar anggota dimensi-dimensi tersebut. Nol, jika tidak ada relasi antar anggota dimensi-dimensi tersebut.

Sebagai contoh adalah shift (S) dan kelas (K), dimana $S = \{\text{Pagi, Siang, Malam}\}$ dan $K = \{1.1, 2.3\}$. Dalam notasi *array*, S dituliskan sebagai:

- $S[0] = \text{Pagi}$
- $S[1] = \text{Siang}$
- $S[2] = \text{Malam}$,

dan K dituliskan sebagai:

- $K[0] = 1.1$
- $K[1] = 2.3$.

Maka hubungan antara shift (S) dan kelas (K), seperti contoh di atas untuk $KS[kelas][shift]$, dapat dituliskan dalam *array* sebagai:

- $KS[0][0] = 0 = KS[1.1][\text{Pagi}]$
- $KS[0][1] = 0 = KS[1.1][\text{Siang}]$
- $KS[0][2] = 1 = KS[1.1][\text{Malam}]$
- $KS[1][0] = 1 = KS[2.3][\text{Pagi}]$
- $KS[1][1] = 0 = KS[2.3][\text{Siang}]$

- $KS[1][2] = 0 = KS[2.3][\text{Malam}]$.

Data Tambahan

Selain itu, ada data-data tambahan, yang bukan merupakan dimensi baru. Data-data ini melekat pada dimensi-dimensi tertentu. Data tambahan tersebut adalah:

1. Kredit (C), adalah jumlah kredit yang harus dipenuhi oleh tiap materi pelajaran yang ada, misal:

- Web Programming adalah 4 kredit
- Decision Support adalah 3 kredit
- Dan seterusnya.

Dari contoh ini, maka secara notasi *array*, C dapat dituliskan sebagai:

- $C[0] = 4$
- $C[1] = 3$,

dimana indeks dari data C ini mengikuti nilai indeks dari himpunan materi (M) yang bersesuaian.

Variabel Solusi

Kemudian didefinisikan sebuah variabel, bernama X, yang memiliki dimensi sejumlah jenis data-data dasar tersebut di atas. Tiap dimensi ini memiliki indeks sejumlah data dari dimensi yang bersangkutan. Sehingga variabel X ini dapat dituliskan dengan notasi *array* sebagai:

$$X[\text{Shift}][\text{Hari}][\text{Jam}][\text{Tingkat}][\text{Kelas}][\text{Materi}][\text{Guru}]$$

atau:

$$X[S][H][J][T][K][M][G].$$

Variabel X ini memiliki nilai 1 (satu) atau 0 (nol). Satu, jika ada relasi antar anggota dimensi-dimensi tersebut. Nol, jika tidak ada relasi antar anggota dimensi-dimensi tersebut. X yang bernilai 1 inilah yang akan menjadi solusi dari penjadwalan perkuliahan.

Variabel X ini disyaratkan harus memenuhi seluruh batasan-batasan pada hubungan antar variabel sebelumnya sebelumnya. Implementasi dari kata “memenuhi” ini artinya, bahwa variabel X bernilai lebih kecil atau sama dengan hubungan antar variabel yang bersesuaian. Sebagai contoh: sebuah data pada Kelas-Shift (KS), misalkan Kelas k harus diajarkan pada Shift s, maka variabel X harus memenuhi syarat: $X[S_s][H][J][T][K_k][M][G] \leq KS[k][s]$.

Dengan cara seperti ini, dapat dijamin bahwa variabel X akan selalu konsisten pada batasan-batasan yang ditetapkan. Perlu ditekankan bahwa tanda yang diberikan adalah “lebih kecil atau sama dengan” (\leq), bukan “sama dengan” ($=$). Cara pembatasan ini berlaku untuk semua relasi dimensi.

Syarat Solusi

Berdasarkan semua penjelasan di atas, maka ditetapkan syarat-syarat (atau *constraints*) berikut ini:

1. X adalah bernilai integer, yang nilainya adalah 0 (nol) atau 1 (satu), dirumuskan sebagai:

$$0 \leq X[s][H][J][T][K][M][G] \leq 1, X \in \text{int} \quad (1).$$
2. Untuk tiap nilai X, harus lebih kecil atau sama dengan data di relasi Guru-Materi (GM) yang bersesuaian, dirumuskan sebagai:

$$X[s][H][J][T][K][M][G_g] \leq GM[g][m] \quad (2).$$
3. Untuk tiap nilai X, harus lebih kecil atau sama dengan data di tabel Kelas-Shift (KS) yang bersesuaian, dirumuskan sebagai:

$$X[s_s][H][J][T][K_k][M][G] \leq KS[k][s] \quad (3).$$
4. Untuk tiap nilai X, harus lebih kecil atau sama dengan data di tabel Kelas-Tingkat (KT) yang bersesuaian, dirumuskan sebagai:

$$X[s][H][J][T_t][K_k][M][G] \leq KT[k][t] \quad (4).$$
5. Untuk tiap nilai X, harus lebih kecil atau sama dengan data di tabel Materi-Tingkat (MT) yang bersesuaian, dirumuskan sebagai:

$$X[s_s][H][J][T][K][M_m][G] \leq MT[m][t] \quad (5).$$
6. Untuk tiap nilai X, harus lebih kecil atau sama dengan data di tabel Guru-Shift -Hari-Jam (GSHJ) yang bersesuaian, dirumuskan sebagai:

$$X[s_s][H_h][J_j][T][K][M][G_g] \leq GSHJ[g][s][h][j] \quad (6).$$
7. Suatu kelas pada waktu tertentu hanya boleh ada satu transaksi perkuliahan. Dengan kata lain, jumlah nilai X untuk kelas yang tertentu dan pada waktu yang tertentu, harus lebih kecil atau sama dengan 1 (satu), dirumuskan sebagai:

$$\sum_{img} X[s_s][H_h][J_j][T][K_k][M][G] \leq 1 \quad (7).$$
8. Suatu dosen pada waktu tertentu hanya boleh ada satu transaksi perkuliahan. Dengan kata lain, jumlah nilai X untuk dosen yang tertentu dan pada waktu yang tertentu, harus lebih kecil atau sama dengan 1 (satu), dirumuskan sebagai:

$$\sum_{tkm} X[s_s][H_h][J_j][T][K][M][G_g] \leq 1 \quad (8).$$
9. Jumlah nilai X untuk materi yang tertentu, harus sama dengan data di tabel Kredit (C) untuk materi tersebut, dirumuskan sebagai:

$$\sum_{shjkg} X[s][H][J][T][K][M_m][G] = C[m] \quad (9).$$

Uji Coba Model

Pengujian dilakukan atas pemodelan di atas, dengan menggunakan sejumlah set data yang tertentu untuk setiap dimensi dan relasi dimensi. Dengan demikian, bisa dicek kebenaran dari hasil perhitungan yang dilakukan.

Detail dari baris-baris pengkodean atau kodifikasi bahasa Java yang penulis lakukan dalam uji coba pemodelan ini dengan menggunakan Choco, tidak dimasukkan ke dalam makalah ini. Ini karena kode yang dibuat untuk uji coba pemodelan ini cukup panjang.

Adapun langkah-langkah algoritmis yang penulis lakukan dalam memasukkan set data contoh ke dalam Choco adalah:

1. Mendefinisikan data dimensi-dimensi dasar, yaitu:
 - a. Shift (S)
 - b. Kelas (K)
 - c. Jam (J)
 - d. Guru (G)
 - e. Materi (M)
 - f. Tingkat (T)
 - g. Hari (H).
2. Mendefinisikan data relasi-relasi antar dimensi yang ada, yaitu:
 - a. Guru-Materi (GM)
 - b. Kelas-Shift (KS)
 - c. Kelas-Tingkat (KT)
 - d. Materi-Tingkat (MT)
 - e. Guru-Shift-Hari-Jam (GSHJ).
3. Memasukkan variabel solusi, yaitu X, ke dalam Choco.
4. Memasukkan semua batasan yang ada ke dalam Choco, seperti yang telah dijelaskan sebelumnya.
5. Propagasi persamaan, yaitu memerintahkan Choco agar mereduksi domain-domain yang tidak perlu dimasukkan dalam perhitungan solusi. Ini perlu dilakukan, agar saat proses pencarian solusi Choco semakin cepat.
6. Pencarian solusi, yaitu memerintahkan Choco untuk mencari nilai solusi pertama yang didapatkan.
7. Pengambilan hasil solusi yang diperoleh dari Choco.

HASIL DAN PEMBAHASAN

Solusi yang didapatkan dari Choco untuk pemodelan di atas untuk data yang relatif sedikit, hanya membutuhkan waktu beberapa detik saja. Beberapa kali percobaan dilakukan untuk sejumlah set data yang berbeda, waktu yang diperlukan relatif sama.

Untuk set data yang sama, hasil yang diperoleh untuk solusi pertama dari setiap kali percobaan, akan menghasilkan data solusi pertama yang selalu sama. Ini terjadi karena secara internal, Choco menerapkan metode numerik yang baku algoritmanya, dan tidak ada proses *heuristic* atau *random generator* di dalamnya.

Choco bisa diperintahkan untuk melakukan pencarian seluruh set solusi yang memungkinkan. Namun, bila pemodelan sistemnya semakin banyak dari sisi dimensi, atau jumlah data, atau domain solusinya, atau tingkat kerumitan persyaratan yang semakin tinggi, maka proses pencarian solusi akan berlangsung semakin lama.

Ini bukan berarti Choco tidak mendapatkan solusinya. Melainkan karena Choco menyimpan dulu solusi yang telah diduplikatnya, dan menampilkan seluruh solusi ketika semua solusi sudah didapatkan. Tapi ini bisa dikendalikan dengan pemrograman berbasis *threading*, yang menghentikan Choco sesaat

ketika solusi didapatkan dan diambil, sebelum dilanjutkan lagi pencarian solusi selanjutnya.

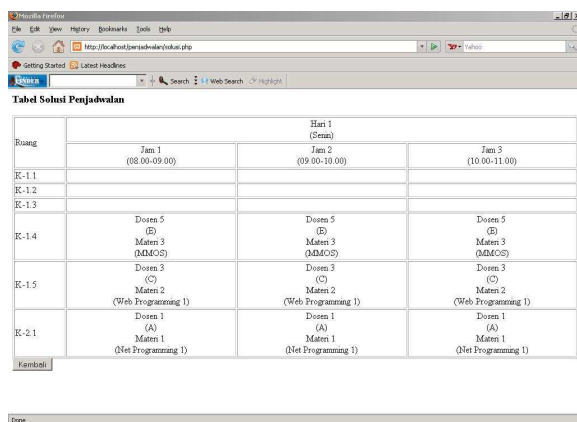
Makalah ini tidak ditujukan membahas secara mendalam dari sisi *performance* atau kinerja, melainkan kepada pemodelan sistem dan pencarian solusinya, dengan menggunakan *Constraint Programming*. Dengan demikian, penulis tidak memberikan perbandingan atau *benchmarking* atas metode-metode lain untuk masalah penjadwalan.

Sampai di sini, penulis telah berhasil membuktikan dan memberikan suatu pemodelan dan simulasi, untuk penyelesaian masalah penjadwalan perkuliahan yang selalu menjadi masalah bagi institusi pendidikan.

Implementasi

Penulis sedang menerapkan pemodelan dan simulasi ini ke dalam aplikasi berbasis PHP (yang terintegrasi dengan Java), ke dalam sistem informasi akademik di intitusi tempat penulis bekerja. Sejauh ini, aplikasi ini bisa bekerja dengan baik dan menunjukkan kinerja yang baik.

Pada implementasi ini, pemodelannya ada beberapa penyederhanaan, yaitu dengan membuang dimensi Shift dan Tingkat. Gambar 1 berikut ini adalah contoh berupa *screenshot* dari implementasi CSP ini, setelah solusi pertama didapatkan.



Kelas	Hari 1 (Senin)		
	Jan 1 (08:00-09:00)	Jan 2 (09:00-10:00)	Jan 3 (10:00-11:00)
K-11			
K-12			
K-13			
K-14	Dosen 5 (B) Matern 3 (MAMC-3)	Dosen 5 (B) Matern 3 (MAMC-3)	Dosen 5 (B) Matern 3 (MAMC-3)
K-15	Dosen 3 (C) Matern 2 (Web Programming 1)	Dosen 3 (C) Matern 2 (Web Programming 1)	Dosen 3 (C) Matern 2 (Web Programming 1)
K-21	Dosen 1 (A) Matern 1 (Net Programming 1)	Dosen 1 (A) Matern 1 (Net Programming 1)	Dosen 1 (A) Matern 1 (Net Programming 1)

Gambar 1. Hasil Solusi Penjadwalan

SIMPULAN

Constraint Programming dapat diterapkan untuk menyelesaikan masalah penjadwalan kuliah dengan mudah dan cepat. Pemodelan dilakukan dengan mudah, sebab yang perlu dimodelkan hanyalah batasan-batasan dari masalah saja, dan tidak perlu memodelkan solusinya, sebagaimana yang biasa terjadi pada *Operation Research*.

Dari sisi praktik, implementasi ke dalam kode pemrograman juga mudah dan cepat. Penerapan batasan-batasan ke dalam kode itu juga mudah untuk dilakukan penyesuaian-penyesuaian yang dibutuhkan, sesuai dengan kondisi di lapangan. Kemudian teknis perhitungannya kita serahkan kepada komponen-komponen yang telah ada.

Keuntungan lainnya, karena pemodelannya yang mudah, maka *debugging* atau pelacakan kesalahan logika juga dapat ditelusuri dan diperbaiki dengan cepat. Selain itu, dengan model implementasi seperti ini, kode dapat digunakan secara berulang-ulang dan dapat digabungkan atau diintegrasikan dengan sistem lainnya.

Pemodelan dalam makalah ini bukanlah sebuah pemodelan akhir yang sempurna untuk masalah penjadwalan perkuliahan, karena pemodelan sistem yang benar-benar tuntas tidak akan pernah ada. Pemodelan sistem akan terus berkembang, sebagaimana sistem itu sendiri berkembang.

Harapan penulis adalah, semoga makalah ini bisa memberikan sedikit kontribusi dan penstimulus dalam dunia pemodelan sistem dan riset operasi (*operation research*) di Indonesia. Dari semua hal yang dibahas dalam efektivitas dan efisiensi, sesungguhnya sangat bergantung kepada pemodelan sistem dan riset operasi.

Akhir dari semuanya, makalah ini masihlah jauh dari sempurna. Untuk itu, penulis mengharapkan kontribusi dan saran dari para pembaca, untuk memberikan masukan kepada penulis demi perbaikan dan kesempurnaan makalah ini.

RUJUKAN

- Barták, R. 1999. *Constraint Programming: In Pursuit of the Holy Grail. Proceedings of the Week of Doctoral Students (WDS99), Part IV*. 1999. Prague. Pp.555-564.
- Barták, R. 1998. *On-Line Guide to Constraint Programming*. (Online). (<http://kti.mff.cuni.cz/~bartak/constraints/>, diakses 27 Agustus 2008).
- Choco. 2007. *Choco: Constraint Programming System*. (Online). (<http://choco-solver.net>, diakses 27 Agustus 2008).
- Abdennadher, S. 2001. *Rule-Based Constraint Programming: Theory and Practice*. München: Ludwig-Maximilians-Universität.
- Abdennadher S., Krämer E., Saft M. dan Schmauss M. 2002. *JACK: A Java Constraint Kit*. (Online). (<http://www.cs.kuleuven.ac.be/~dtai/projects/ALP/newsletter/feb02/nav/monfroy/index.html>, diakses 27 Agustus 2008).
- Wikipedia. 2008. *Constraint Programming*. (Online). (http://en.wikipedia.org/wiki/Constraint_programming, diakses 27 Agustus 2008).