APLIKASI PENJADWALAN PERKULIAHAN MENGGUNAKAN ALGORITMA *SEQUENTIAL SEARCH* DAN *FORWARD CHECKING*

Eduardus Hardika Sandy Atmaja[1)], **Eko Hari Parnadi**[2)]

[1), 2)] *Teknik Informatika Universitas Sanata Dharna Yogyakarta*
Kampus III Paingan,Maguwoharjo, Depok Sleman, Yogyakarta 55282
Email : eduardus777@gmail.com[1)], *harimbi.parmadi@gmail.com*[2)]

Abstrak

Penyusunan jadwal perkuliahan secara manual selain memakan banyak waktu juga membutuhkan ketelitian agar tidak terjadi tabrakan jadwal. Banyaknya kombinasi jadwal dan berbagai constraint yang diberikan, antara lain : dosen yang memiliki jabatan struktural tidak dapat dijadwalkan pada waktu tertentu, setiap mahasiswa pada semester yang sama tidak boleh kuliah lebih dari dua sesi semakin menambah rumit dalam penyusunan jadwal. Selain itu, penyusunan jadwal juga harus mempertimbangkan prioritas untuk Mata Kuliah Pengembangan Kepribadian (MPK) yang dikelola oleh Unit Penyelenggaran Mata Kuliah Pengembangan Kepribadian (UP MPK).

Aplikasi penjadwalan mata kuliah ini dirancang dengan mempertimbangkan ketersediaan hari, sesi dan ruangan serta mengakomodasi constraint yang diberikan. Dengan demikian jadwal yang dihasilkan menjadi lebih baik dan tidak memakan banyak waktu jika dibandingkan dengan cara manual. Algoritma sequential search dimanfaatkan untuk mencari kombinasi jadwal yang tidak bertabrakan serta disimpan dalam bentuk array. Sedangkan forward checking digunakan untuk mengecek semester pada jadwal yang telah terbentuk perhari.

Hasil dari penelitian ini menunjukkan bahwa aplikasi penjadwalan menggunakan algorima forward checking dan sequential search mampu menghasilkan efisiensi pemakaian ruang sebesar 80% sedangkan dengan cara manual, efisiensi pemakaian ruang hanya sebesar 71,4%. Aplikasi ini juga efektif, dalam arti tidak ada jadwal yang bertabrakan satu sama lain.

Kata kunci : *penjadwalan, forward checking, sequential search, constraint, efisiensi.*

1. Pendahuluan

Penjadwalan mata kuliah merupakan proses penyusunan jadwal perkuliahan yang memerlukan tingkat ketelitian yang tinggi agar tidak terjadi tabrakan jadwal baik untuk dosen yang mengampu mata kuliah maupun ruangan yang digunakan untuk perkuliahan. Proses penyusunan jadwal di jurusan Teknik Informatika saat ini masih menggunakan cara manual sehingga memakan banyak waktu akibat banyaknya kombinasi jadwal. Hal ini mengakibatkan sering adanya tabrakan jadwal. Penyusunan jadwal juga semakin rumit jika terdapat perubahan jadwal karena ada dosen yang tidak dapat mengajar pada jadwal yang telah ditentukan. Akibatnya penyusunan jadwal secara keseluruhan menjadi berubah.

Kerumitan penyusunan jadwal perkuliahan juga dipengaruhi oleh banyaknya dosen jurusan Teknik Informatika yang memiliki jabatan struktural sehingga pada hari dan jam tertentu tidak dapat mengisi perkuliahan karena harus mengikuti rapat kerja ataupun tugas lain yang berkaitan dengan jabatan struktural mereka. Terbatasnya ruang kuliah yang digunakan semakin menambah kerumitan penyusunan jadwal. Hal ini mengakibatkan penyusunan jadwal menjadi sulit jika menggunakan cara manual. Melihat permasalahan tersebut, diperlukan sistem penyusun jadwal yang secara otomatis dapat menghasilkan kombinasi jadwal kuliah yang efektif dan memenuhi *constraint*. Efektif yang dimaksud adalah tidak ada jadwal yang bertabrakan satu dengan yang lainnya. C*onstraint* yang dimaksud adalah jadwal yang diprioritaskan bagi dosen yang memiliki jabatan struktural dan bagi dosen yang berhalangan hadir pada waktu tertentu. Mata kuliah MPK dan jumlah maksimal perkuliahan untuk semester yang sama dalam satu hari juga turut diperhatikan.

Masalah yang akan diselesaikan dalam penelitian ini adalah mencari kombinasi mata kuliah yang sesuai *constraint* untuk menghasilkan jadwal perkuliahan yang efektif dengan menggunakan algoritma *forward checking* dan *sequentialsearch.*

Tujuan yang ingin dicapai dari penelitian ini adalah (1) Mengetahui efektifitas algoritma *forward checking* dan *sequential search* dalam menghasilkan jadwal perkuliahan. (2) Memudahkan penyusunan jadwal ulang jika terjadi perubahan.

Metode yang digunakan dalam penelitian ini antara lain : (1) Pengumpulan data, dilakukan dengan wawancara langsung kepada sekretaris jurusan selaku penyusun jadwal serta melihat data jadwal kuliah yang telah dibuat

sebelumnya. (2) Perancangan sistem, membuat *use case*, DFD (*Data Flow Diagram*), algoritma, diagram kelas, desain basis data dan rancangan antarmuka. (3) Implementasi sistem menggunakan *Oracle* dan *Java*. (4) Pengujian sistem untuk melihat keefektifan algoritma dalam menemukan jadwal sesuai dengan *constraint*. Uji coba sistem juga dilakukan kepada sekretaris jurusan Teknik Informatika untuk mengetahui kesalahan yang ada pada sistem. (5) Analisis sistem untuk melihat efektifitas dan efisiensi penggunaan ruangan.

Tinjuan pustaka dalam penelitian ini antara lain :
Permasalahan penjadwalan perkuliahan telah dicoba untuk diteliti oleh dari banyak peneliti. Sejumlah metode telah dihasilkan untuk mendapatkan jadwal yang optimum. Permasalahan penjadwalan pengajaran klasik didefinisikan sebagai berikut [1]: Terdapat sejumlah m kelas $\{c_1, ..., c_m\}$, n guru $\{t_1 ..., t_n\}$ dan p periode {1, ..., p}. Terdapat pula matriks bilangan bulat tak negatif R_{mn}, yang disebut matriks *requirements*, dengan ri_{ij} adalah jumlah pelajaran yang diberikan oleh guru t_j pada kelas c_i.

Masalah penjadwalan juga pernah dimodelkan menggunakan pendekatan CSP dengan teknik klasik pencarian *backtracking* [2]. Pencarian ini menggunakan *backtracking* mempunyai beban komputasi yang agak berat. Untuk lebih meringankan beban komputasi beberapa peneliti menggabungkan dengan algoritma genetika [3] dengan hasil waktu komputasi yang lebih baik. Beberapa peneliti juga melakukan variasi untuk menggunakan algoritma algoritma yang lain (algoritma *hill climbing*). Masih banyak peluang untuk perbaikan efisiensi algoritma dengan algoritma-algoritma *hybrid*/campuran [4].

Sebuah *constraint satisfaction problem* direpresentasikan dengan tiga himpunan variabel yaitu Z, D dan C. <Z, D, C >. Z adalah himpunan dari variabel dengan jumlah tertentu x1, x2.....xn. D adalah domain sebuah fungsi yang memetakan setiap variabel pada Z ke sebuah himpunan dari obyek. D: Z → Himpunan obyek berhingga. Dz sebagai himpunan obyek yang dipetakan dari x_1 oleh D. Obyek ini adalah semua nilai yang mungkin bagi x_1 oleh himpunan Dx sebagai domain xi. C adalah sebuah himpunan *constraint* yang berhingga atas sebuah subset variabel-variabel di Z. C adalah sebuah kumpulan label. $C_{x1,x2.....xk}$ membatasi relasi himpunan $_{x1,x2...dan\ xk}$ yang dapat diambil secara bersamaan, misalkan variabel x mempunyai domain D (a,b,c) , dan misalnya *constraint* C adalah C_{x1} = (<x1,a>,<x1,b>,<x1,c>). C_x adalah sebuah himpunan kumpulan label sementara D_x adalah sebuah himpunan nilai. Nilai x yang mungkin juga harus memenuhi kombinasi *constraint* C yang lain ($C_{x2}\ldots_{xk}$).

Sebuah tupel solusi atas sebuah *constraint satisfaction problem* adalah kumpulan label yang semua anggotanya memenuhi semua *constraint*.

$$\forall csp((Z,D,C)) : \forall x_1 x_2 ,.....x_n \in Z :$$

$$\forall v_1 \in D_{x1}, v_2 \in D_{x2,...,} v_n \in D_{xn}$$

Tupel solusi ((<x_1,v_1> <x_2,v_2>....<x_n,v_n>, (Z,D,C)) ≡ ((Z = x_1,x_2,...x_n) ∧ (∀ C ∈ *C* : memenuhi *constraint* ((x_1, v_1><x_2, v_2>…<x_n, v_n>), C)))

Penyelesaian CSP dapat dirumuskan sebagai berikut: (1) Pemberian nilai yang tidak melanggar sembarang *constraint* (konsisten). (2) Pemberian nilai secara lengkap, apabila setiap variabel diberi nilai. (3) Penyelesaian CSP adalah pemberian nilai yang konsisten dan lengkap.

Sebuah *constraint problem* disebut memiliki penyelesaian jika memiliki tupel solusi. Tergantung dari kebutuhan aplikasinya, *constraint satisfaction problem* dapat dikategorikan menjadi beberapa kategori sebagai berikut : (1) Pencarian solusi terbatas. (2) Pencarian semua solusi. (3) Pencarian solusi yang optimal, optimal sesuai pendefinisian

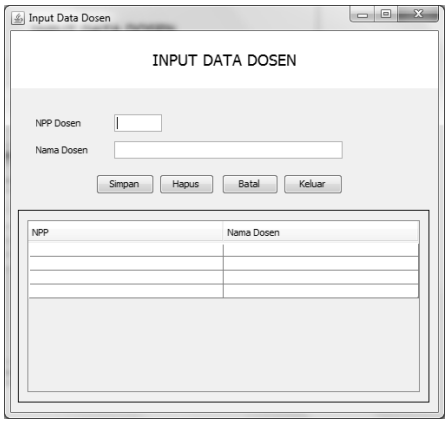
Penyelesaian CSP perlu didefinisikan *state space* (ruang keadaan) dari masalah tersebut, kemudian tiap-tiap *state* dalam CSP didefinisikan oleh sebuah pemberian nilai *(assignment)* nilai ke beberapa atau semua variabel. Setelah ruang *state* atau model matematika dijabarkan kemudian solusi dari CSP dapat dihasilkan dengan proses pencarian.

Metode *Sequential Search* atau disebut pencarian beruntun dapat digunakan untuk melakukan pencarian data baik pada *array* yang sudah terurut maupun yang belum terurut. Proses yang terjadi pada metode pencarian ini adalah sebagai berikut [5] : (1) Membaca *array* data. (2) Menentukan data yang dicari. (3) Mulai dari data pertama sampai dengan data terakhir, data yang dicari dibandingkan dengan masing-masing data di dalam *array*. Jika data yang dicari tidak ditemukan maka semua data atau elemen *array* dibandingkan sampai selesai. Jika data yang dicari ditemukan maka perbandingan akan dihentikan

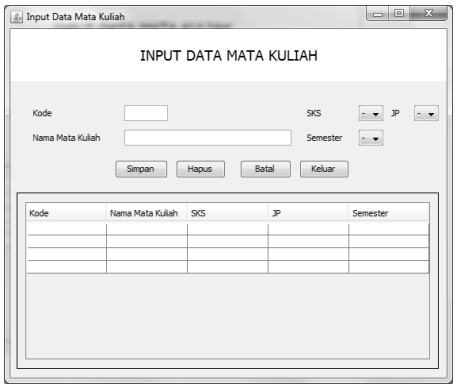
2. Pembahasan

Algoritma yang digunakan dalam pembuatan sistem penjadwalan ini adalah *forward checking* dan *sequential search. Sequential search* berfungsi untuk mencegah adanya 2 dosen mengajar pada sesi yang sama dengan melakukan pencocokan calon jadwal dengan setiap jadwal pada setiap sesi. *Forward checking* berfungsi untuk mengecek semester pada jadwal yang terbentuk perhari yang sudah dimasukkan ke dalam pohon.

Alur penyusunan jadwal dimulai dengan memasukkan semua kesanggupan mengajar (data dosen dan mata kuliah yang diampu) ke dalam *list* jadwal.

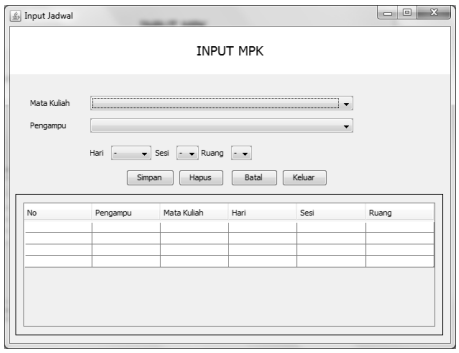


Gambar 1. *Input Data Dosen*

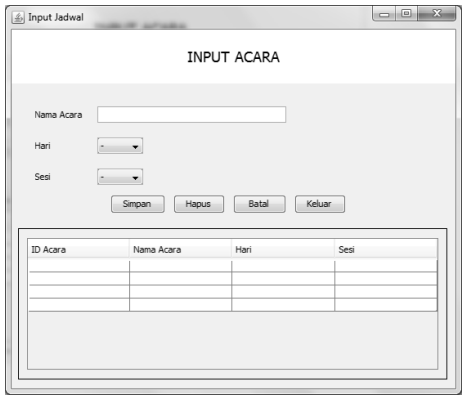


Gambar 2. *Input Data Mata Kuliah*

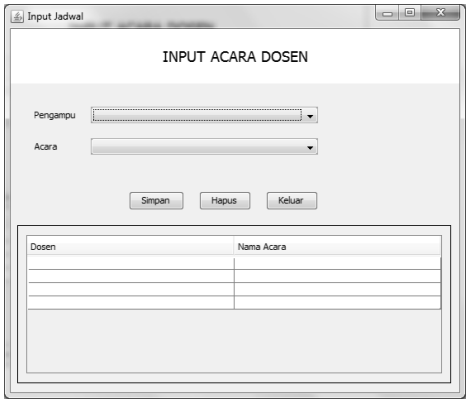
Gambar 3. *Input Data Ruang*



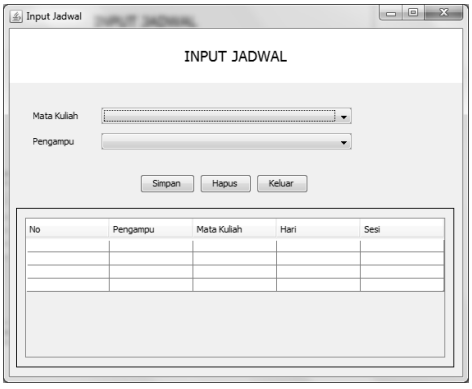
Gambar 4. *Input Data MPK*



Gambar 5. *Input Acara*



Gambar 6. *Input Constraint Dosen*



Gambar 7. *Input Kesanggupan Mengajar*

Sistem akan menyimpan semua inputan ke dalam *list* jadwal dan akan dimasukkan ke dalam matriks 3 x n. Tiga merupakan banyak sesi dan n merupakan banyak ruang yang tersedia.

	1	2	...	n
Sesi 1	d,m			
Sesi 2				
Sesi 3				

d adalah dosen
m adalah mata kuliah

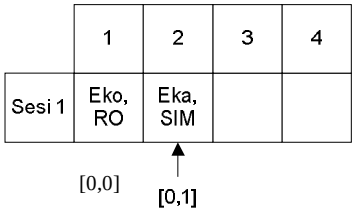
Gambar 8. *Matriks 3xn*

Sebelum masing-masing kesanggupan mengajar masuk ke dalam matriks, obyek (pasangan dosen dan mata kuliah) tersebut akan diberi hari, sesi sementara (calon jadwal) serta akan dicek terlebih dahulu. Pengecekan tersebut meliputi pengecekan MPK, pengecekan tabrakan jadwal, *constraint* mengajar dan pengecekan semester. Jika keempat pengecekan tersebut terpenuhi maka obyek tersebut akan menjadi jadwal.

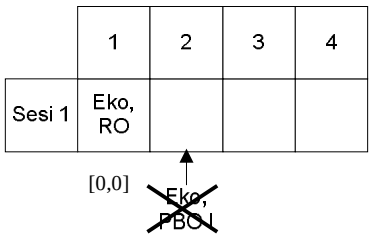
Pengecekan yang pertama adalah Mata Kuliah Pengembangan Kepribadian (MPK) yang sudah ditentukan terlebih dahulu hari dan sesinya. Setiap calon jadwal yang masuk akan dicek terlebih dahulu hari dan sesinya. Jika hari dan sesi calon jadwal sama dengan hari dan sesi salah satu MPK maka calon jadwal tersebut akan diabaikan. Jika calon jadwal tidak sama dengan hari dan sesi salah satu MPK maka calon jadwal tersebut akan masuk ke dalam pengecekan yang kedua.

Pengecekan yang kedua adalah pengecekan tabrakan jadwal dengan memanfaatkan algoritma *sequential search* untuk menghindari adanya dosen yang sama pada satu sesi yang sama.

Sequential search : Jika matriks pada baris sesi 1 masih kosong maka obyek pertama akan dimasukkan pada indeks [0][0]. Selanjutnya obyek kedua akan dibandingkan dengan obyek pertama dengan membandingkan nama dosen. Jika obyek kedua sama dengan obyek pertama maka obyek kedua tidak akan masuk matriks. Jika obyek kedua berbeda maka obyek kedua akan masuk ke dalam posisi [0][1]. Ulangi langkah tersebut hingga ruang habis dan lanjut ke sesi berikutnya.



Gambar 9. *Obyek Masuk*



Gambar 10. *Obyek Tidak Masuk*

Matriks 3 x n yang sudah terisi menunjukkan jadwal untuk satu hari, selanjutnya akan dibentuk matriks baru untuk hari berikutnya dan ulangi langkah pengecekan.

	1	2	3	4
Sesi 1	Eko, RO	Eka, SIM	Tatik, PBO II	Albert, PBO II
Sesi 2	Bram, IMK	Linggo, Citra	Rosa, SD I	Wawan, SD I
Sesi 3	Polina, PTI	Puspa, PTI	Iwan, PTI	Wawan, SD I

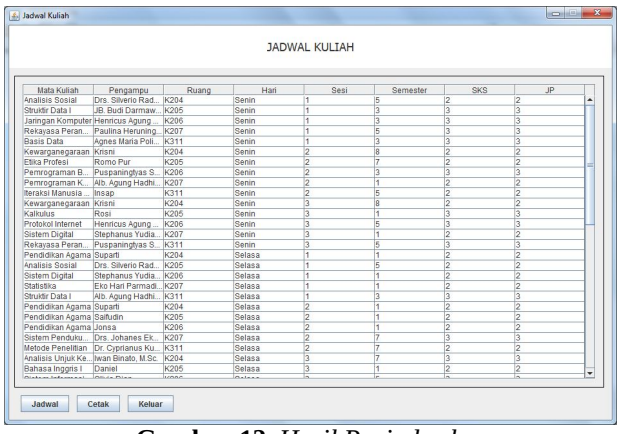
Gambar 11. *Jadwal Hari Pertama*

Pengecekan yang ketiga adalah *contraint* mengajar. Pengecekan ini dilakukan setiap *sequential search* selesai membandingkan satu indeks. Setiap calon jadwal yang masuk akan dicek terlebih dahulu acara yang diikuti oleh dosen pengampunya. Jika hari dan sesi calon jadwal sama dengan hari dan sesi salah satu acara dosen pengampunya maka calon jadwal tersebut akan diabaikan. Jika hari dan sesi calon jadwal tidak sama dengan hari dan sesi salah satu acara dosen pengampunya maka calon jadwal tersebut akan di-*update* ke *database* untuk menambahkan hari, sesi dan ruang kuliah sesuai dengan posisi pada matriks.

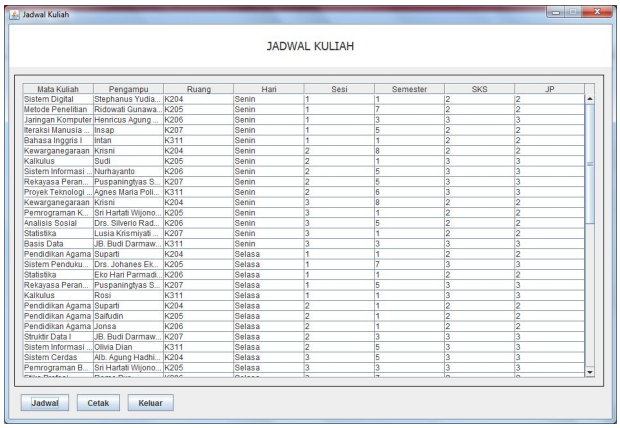
Saat obyek di-*update* obyek tersebut juga akan dimasukkan ke dalam pohon. Setiap pohon merupakan representasi dari jadwal untuk perharinya. Obyek yang pertama kali masuk akan dijadikan sebagai *root*. Selanjutnya obyek yang baru akan dibandingkan ID jadwalnya. Jika ID jadwal lebih kecil dari ID *root* maka obyek akan menjadi anak kiri. Jika ID jadwal lebih besar dari ID *root* maka obyek akan menjadi anak kanan. Ulangi langkah tersebut hingga jadwal untuk satu hari selesai.

Setelah pohon selesai dibentuk untuk perharinya selanjutnya dilakukan pengecekan yang keempat yaitu pengecekan semester. Setiap jadwal yang masuk akan dicek terlebih dahulu semester mata kuliahnya. Jika pada hari yang sama dan sesi yang berbeda jumlah mata kuliah dengan semester yang sama adalah tiga maka jadwal tersebut akan dihapus dari jadwal dan akan kembali menjadi calon jadwal. Jika pada hari yang sama dan sesi yang berbeda jumlah mata kuliah dengan semester yang sama kurang dari tiga maka jadwal tersebut akan diabaikan. Proses pengecekan dilakukan dengan memanfaatkan *forward checking.*

Pengecekan dimulai dari *root*, setelah itu akan bergerak ke anak kiri. Lakukan pengecekan hingga tidak terdapat lagi anak kiri, setelah mencapai daun anak kiri pengecekan dilanjutkan ke anak kanan pada *node* sebelumnya. Ulangi langkah tersebut untuk semua *node* pada pohon. Setelah dilakukan pengecekan, jadwal telah terbentuk dan siap ditampilkan.



Gambar 12. *Hasil Penjadwalan*



Gambar 13. *Alternatif jadwal*

Apabila sekretaris jurusan masih menghendaki alternatif jadwal lain maka sekretaris cukup memilih tombol jadwal. Selanjutnya sistem akan mengacak kembali dan diperoleh jadwal baru.

Tabel 1. *Tabel* penggunaan ruang dengan cara manual

No.	Hari	Mata Kuliah	Ruang	Total slot	Sisa Slot
1	Senin	14	5	15	1
2	Selasa	16	8	24	8
3	Rabu	9	5	15	6
4	Kamis	12	5	15	3
5	Jumat	9	5	15	6
	Jumlah	60		84	24

Tabel 1. *Tabel* penggunaan ruang dengan aplikasi

No.	Hari	Mata Kuliah	Ruang	Total slot	Sisa Slot
1	Senin	15	5	15	0
2	Selasa	15	5	15	0
3	Rabu	15	5	15	0
4	Kamis	14	5	15	1
5	Jumat	1	5	15	14
	Jumlah	60		75	15

Selanjutnya efisiensi penggunaan ruang dihitung berdasarkan persamaan (1) di bawah ini :

$$\text{efisiensi} = \frac{\text{jumlah mata kuliah}}{\text{total slot}} \times 100\% \quad \ldots\ldots\ldots\ldots (1)$$

$$\text{efisiensi(manual)} = \frac{60}{84} x\ 100\% = 71{,}4\%$$

$$\text{efisiensi(aplikasi)} = \frac{60}{75} x\ 100\% = 80\%$$

3. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini antara lain:

1. Sistem penjadwalan kuliah meningkatkan efisiensi penggunaan ruangan jika dibandingkan dengan cara manual.
2. Algoritma *forward checking* dan *sequential search* dapat membantu menyelesaikan kasus penjadwalan dengan beberapa *constraint* yang diberikan.

Saran yang dapat diberikan oleh penulis antara lain :

1. Penelitian dikembangkan dengan penambahan kelas praktikum.
2. Jam perkuliahan disesuaikan dengan kondisi nyata tanpa mengasumsikan dengan menggunakan sesi.

Daftar Pustaka

[1] Werra, D.D., *"An Introduction to Timetabling. European Journal of Operational Research"*, 19(2) 151–162, 1985.
[2] Barták, R., *"On-Line Guide To Constraint Programming"*, http://kti.mff.cuni.cz/~bartak/constraints/, 2007.
[3] Ozcan, E., Alken, A., *"Timetabling using a Steady State Genetic Algorithm.* In: The 4th International Conference on the Practice And Theory of Automated Timetabling", 2002.
[4] Chiarandini, M., Birattari, M., Socha, K., Rossi- Doria, O., *"An Effective Hybrid Algorithm for University Course Timetabling"*, Journal of Scheduling 9(5) (2006) 403–432, 2006.
[5] Utami Ema, dkk., "STRUKTUR DATA : Konsep dan Implementasinya dalam Bahasa C dan *Free Pascal* di GNU/Linux", Yogyakarta : Graha Ilmu, 2007.

Biodata Penulis

Eduardus Hardika Sandy Atmaja adalah mahasiswa Jurusan Teknik Informatika Universitas Sanata Dharma. Saat ini sedang menjalani tahap akhir penyelesaian skripsi untuk memperoleh gelar Sarjana Komputer (S.Kom.).

Eko Hari Parmadi, S.Si., M.Kom., memperoleh gelar Sarjana Sains (S.Si.), pada Jurusan Matematika Fakultas MIPA Universitas Gajah Mada, lulus tahun 1995. Memperoleh gelar Magister Komputer (M.Kom) pada Program Pasca Sarjana Magister Ilmu Komputer Universitas Gajah Mada Yogyakarta, lulus tahun 2001. Saat ini menjadi Dosen tetap di Jurusan Teknik Informatika Universitas Sanata Dharma Yogyakarta.