Pembangkitan Solusi Penjadwalan Berprioritas Melalui Penerapan *Constraint Satisfaction Problem* (Studi Kasus: Laboratorium Fakultas Teknologi Informasi Universitas XXX)

Chandra Ari Gunawan [#1], Hapnes Toba[#2]

#Program Studi Teknik Informatika Universitas Kristen Maranatha
Jl. Suria Sumantri 65, Bandung
[1]feechanz@yahoo.co.id
[2]hapnestoba@it.maranatha.edu

Abstract — **Generating schedule for personnel is one of the classic problems in organizations. Although there are many techniques have been introduced, there is no ultimate solution has been known. In this sense, there are always some aspects which are unique and specific applied in an organization. In this paper we propose an approach to generate personnel schedules in a flexible manner. The flexibility is offers in a way that the administrator can set which criteria should be prioritized, and the personnels are allow to enter their available schedules. The priority level is fulfilled by applying the 'most constraining value'-strategy of the Constraint Satisfaction Problem (CSP) approach. The approach is evaluated in a staff scheduling system at the Computer Laboratory of the Faculty of Information Technology (FIT) at XXX University. The results show that the approach has comparable schedule candidates, comparing to the ones created manually by human.**

Keywords — **Constraint Satisfaction Problem, Rekayasa Pembangkitan Jadwal, Mekanisme Penjadwalan Staf, Sistem Pengelolaan Laboratorium, Pola Desain Perangkat Lunak**

I. PENDAHULUAN

Pembuatan jadwal untuk sebuah organisasi seperti misalnya dalam pengelolaan staf laboratorium merupakan proses yang rutin dilakukan. Berdasarkan pengamatan penulis, di laboratorium FIT Universitas XXX, terdapat beberapa persoalan klasik dalam pembuatan jadwal secara manual, di antaranya adalah:

1. Membutuhkan waktu yang lama untuk memeriksa batasan-batasan yang diajukan oleh orang-orang yang terlibat dalam penyusuan jadwal.
2. Sering terjadi perubahan prioritas dan aturan dalam pembentukan jadwal.

Dalam penelitian ini diusulkan sebuah pendekatan yang dapat melakukan penyusunan jadwal secara otomatis untuk pengelolaan staf. Keistimewaan pendekatan yang diusulkan adalah menawarkan adanya fleksibilitas dalam pendefinisian persyaratan saat pembuatan jadwal dilakukan. Pendekatan tersebut dilakukan dengan menerapkan algoritma CSP. Pemilihan CSP didasari terutama pada konsepnya yang fleksibel untuk diimplementasikan, namun akurat serta cocok dalam domain permasalahan yang tidak terlalu besar, serta cocok pula untuk aturan-aturan berbasis batasan tekstual, seperti dalam pembuatan jadwal [1]-[4]. Jika dibandingkan dengan pendekatan lain yang sering dimanfaatkan dalam optimasi jadwal, seperti algoritma genetika, yang memerlukan adanya penentuan fungsi optimasi (*fitness*), maka dalam CSP kandidat jadwal yang dibangkitkan adalah hasil pembangkitan yang dilakukan secara bertingkat, misalnya berdasarkan prioritas kriteria dalam domain, sehingga diharapkan lebih memenuhi persyaratan yang diinginkan dan fleksibel [4]-[5].

II. KAJIAN LITERATUR

A. Constraint Satisfaction Problem

CSP adalah sebuah pendekatan untuk mencari suatu solusi dari sebuah masalah dengan cara mencari objek atau keadaan yang memenuhi sebuah kriteria atau persyaratan. Sebuah *constraint* diartikan sebagai batasan atau kriteria yang perlu dipenuhi. Terdapat tiga komponen utama yang perlu diperhatikan dalam pendekatan CSP, yaitu [6]-[8]:

1. *Constraint*: merupakan suatu aturan yang ditentukan untuk mengatur nilai yang boleh diisikan ke dalam variabel atau kombinasi variabel. Terdapat beberapa jenis *constraint*, di antaranya *unary* (menyatakan sebuah variabel), *binary* (menyatakan persyaratan sepasang variabel), *n-ary* (menyatakan persyaratan tiga atau lebih variabel), dan *preference* (syarat yang sebaiknya dipenuhi, tetapi tidak harus)

2. Domain: merupakan kumpulan nilai legal yang dapat diisi ke dalam variabel. Dengan kata lain, sebuah domain akan membatasi nilai suatu variabel.
3. Variabel: merupakan suatu penampung yang dapat diisi dengan berbagai nilai. Biasanya persoalan dimulai di sini, yaitu ketika variabel harus diisi oleh domain yang telah memenuhi *constraint*.

B. Jenis Constraint

Secara pemenuhan prioritasnya, *constraint* dapat dibedakan menjadi 2 bagian, yaitu [6]-[8]:

1. *Hard Constraint*: adalah persyaratan yang harus dipenuhi dan tidak boleh dilanggar dalam pembuatan penyelasaian masalah.
2. *Soft Constraint*: adalah persyaratan tambahan yang biasanya merupakan sebuah permintaan khusus, tanpa adanya konsekuensi fatal jika tidak terpenuhi.

C. Menentukan Constraint

Untuk menentukan isi sebuah variabel yang hendak diisi, dapat dilakukan melalui dua cara yaitu [6]-[8]:

1. *Most Constrained Variable*
   Penentuan variabel yang pertama diisi dan berikutnya dimulai dari variabel yang paling banyak mengandung *constraint*.
2. *Least Constrained Variable*
   Penentuan variabel yang pertama diisi dan berikutnya dimulai dari variabel yang paling sedikit mengandung *constraint*.

D. Penerapan CSP dalam Penjadwalan dan Kontribusi Penelitian

Pemanfaatan algoritma berbasis CSP banyak dilakukan untuk mencari solusi dalam hal penjadwalan, misalnya dalam berbagai riset terkini [1]-[5], [10]-[13]. Van den Bergh, dkk dalam [4] menunjukkan tingkat kompleksitas permasalahan penjadwalan personal yang mencakup berbagai hal, seperti: waktu kerja, *shift* kerja, usia, golongan dan pengupahan, preferensi personal dan alokasi kompetensi. Berbagai pendekatan yang banyak digunakan dalam pemecahan problem penjadwalan tersebut tersebar mulai dari metode matematis, heuristis, simulasi, CSP, antrean, dan lainnya. Tidak ada satu metode pun yang dapat menangani keunikan setiap permasalahan. Sebuah pengaplikasian penjadwalan dalam [1], [3], [4]-[5] disarankan untuk dilakukan secara spesifik dalam problem yang dihadapi karena faktor-faktor yang dihadapi sangat tidak pasti. Namun demikian tentu saja dimungkinkan untuk dapat membuat solusi *general* untuk problem-problem yang sejenis. Pendekatan melalui CSP merupakan salah satu pendekatan yang sangat fleksibel. Dari riset-riset terkini dalam bidang penjadwalan, CSP dianggap cocok untuk jumlah variabel dan domain yang tidak terlalu besar, apalagi jika jumlah keduanya dapat dipastikan sejak awal.

Sebuah permasalahan penjadwalan dengan penekanan pada pembagian jumlah jam kerja merata, dilakukan oleh Lapègue, dkk dalam [11]. Konsentrasi pendekatan yang ditawarkan dalam [11] mengedepankan bagaimana seorang pekerja mendapatkan jam kerja yang merata dalam *shift* jam kerja yang telah ditentukan sebelumnya di dalam organisasi. Riset dalam [11] tidak menunjukkan bagaimana seandainya ada prioritas yang diinginkan oleh seseorang, seperti misalnya jika ada jadwal lain yang perlu dihindari. Berbeda dengan [11], dalam konteks prioritas inilah diusulkan dalam makalah yang dituliskan ini, untuk memisahkan antara *hard-* dan *soft constraints*, dengan prioritas pada kondisi-kondisi tertentu, seperti jadwal malam.

Di sisi lain, Hadwan, dkk dalam [10] dan Meskens, dkk dalam [13] mengusulkan adanya optimasi dalam pemenuhan kondisi antara jadwal pribadi dan kebutuhan organisasi. Salah satu isu yang dihadapi dalam proses pencarian optimasi adalah dalam hal menentukan *objective function* yang diinginkan, dan untuk ini diperlukan eksperimen yang mendalam atau dengan memanfaatkan pendekatan matematis. Dalam konteks penjadwalan staf laboratorium, dengan mempertimbangkan bahwa variabel dan domain yang tersedia sudah memberikan kondisi 'ideal' yang diinginkan oleh masing-masing pihak, yaitu pihak laboratorium dan mahasiswa, maka diusulkan kondisi 'ideal' itulah yang perlu disusun sedemikian rupa melalui pemberian prioritas pada *constraint*, tanpa perlu adanya optimasi.

Permasalahan penjadwalan yang mengaitkan antara keinginan pribadi pekerja dengan ketersediaan jam kerja dalam organisasi dibahas dalam [12]. Ásgeirsson dalam [12] mengusulkan agar setiap keinginan pribadi tersebut dijadikan sebagai hal yang dicari solusinya pada tahap terakhir pada saat jadwal diverifikasi oleh tim dalam organisasi. Konsep yang ditawarkan dalam [12] adalah dengan membuat modul-modul yang terpisah antara pembangkitan jadwal awal, perbaikan *under* dan *overstaffing* dalam *shift* jam kerja, dan kemudian verifikasi keinginan pribadi dari pekerja. Hal tersebut dianggap cocok untuk dilakukan dalam konteks jadwal pekerja dalam organisasi dengan jumlah pekerja cukup banyak, karena dapat mengalokasikan jadwal pada pekerja lain.

Dalam konteks penjadwalan staf laboratorium yang dipaparkan dalam makalah ini, jumlah staf yang tersedia tidak cukup banyak dan mahasiswa sebenarnya memiliki jadwal yang relatif lebih stabil dalam kurun satu semester. Oleh karena itu, pendekatan yang dirasakan lebih cocok adalah dengan lebih dahulu mengisikan jadwal-jadwal dengan variabel yang paling banyak batasannya (*most constrained variable*), sehingga variabel lainnya bisa lebih bebas diisikan oleh staf lainnya. Selain itu, dengan membentuk beberapa kandidat penjadwalan, seorang koordinator staf memiliki kemampuan untuk memverifikasi jadwal dengan melibatkan staf yang mengajukan keberatan.

III. ANALISIS DAN PERMODELAN

A. Arsitektur Sistem

Berikut ini adalah arsitektur sistem yang meliputi: *use case diagram*, *activity diagram*, dan pemodelan algoritma CSP.

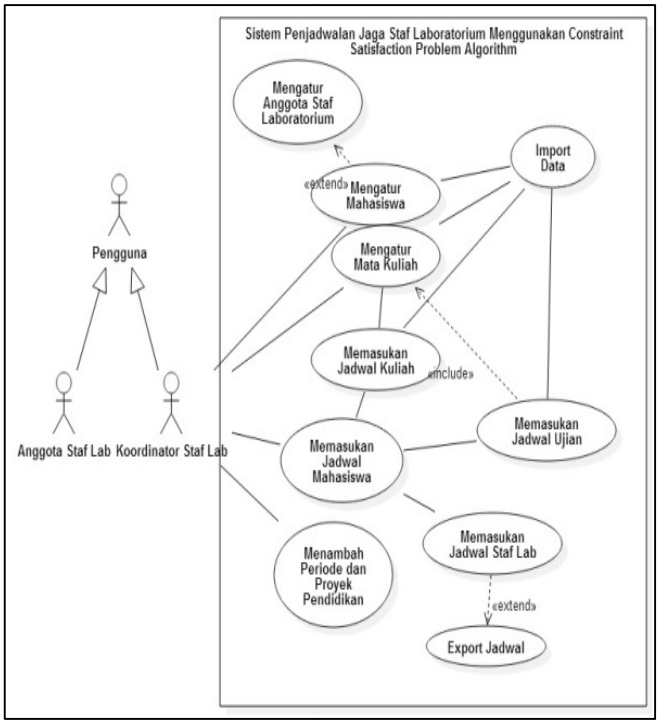
5) Use Case Diagram

Rancangan *use case diagram* secara menyeluruh dapat dilihat pada Gambar 1. Aktor utama dalam sistem ini adalah pengguna aplikasi, yang terbagi ke dalam aktor staf dan koordinator staf. Pengguna dapat mengakses berbagai modul yang disediakan pada aplikasi, yang kemudian dapat melakukan berbagai operasi yang dialokasikan pada dirinya. Koordinator staf berperan penting dalam memberikan masukan untuk berbagai batasan yang telah ditentukan sebelumnya.

6) Activity Diagram

Activity diagram untuk aplikasi dapat dilihat dalam Gambar 2-4. Penjelasan detail hanya diberikan untuk bagian utama yang berhubungan dengan sistem penjadwalan.

1. Memfilter Jadwal dengan CSP
   Setiap mahasiswa memiliki 3 jadwal yang mungkin perlu diperhatikan pada saat pembentukan jadwal jaga. Namun hal ini tergantung daripada masukan untuk menentukan dari ketiga jadwal tersebut, jadwal mana saja yang akan diperhatikan pada saat pembentukan jadwal. Ketiga jadwal tersebut meliputi: jadwal kuliah, jadwal ujian dan jadwal lain-lain, seperti: jadwal kegiatan kemahasiswaan, jadwal asistensi, ataupun jadwal kegiatan non-akademik di luar kampus.
2. Mengatur Jadwal Kuliah Mahasiswa
   Mahasiswa yang bertugas sebagai staf juga memiliki jadwal kuliah yang perlu diperhatikan pada saat membentuk jadwal jaga. Di sisi lain, jadwal ujian perlu juga dihubungkan dengan jadwal kuliah.
3. Mengatur Jadwal Mahasiswa
   Mahasiswa yang bertugas sebagai staf, selain memiliki jadwal kuliah yang mungkin perlu diperhatikan, juga memiliki jadwal mahasiswa. Jadwal mahasiswa adalah jadwal yang tidak memiliki hubungan dengan jadwal kuliah namun mungkin perlu diperhatikan saat membentuk jadwal jaga. Sisi fleksibilitas yang ditawarkan dalam makalah ini ditekankan dalam hal pengaturan jadwal mahasiswa ini, yang dianggap sebagai *soft constraint*.



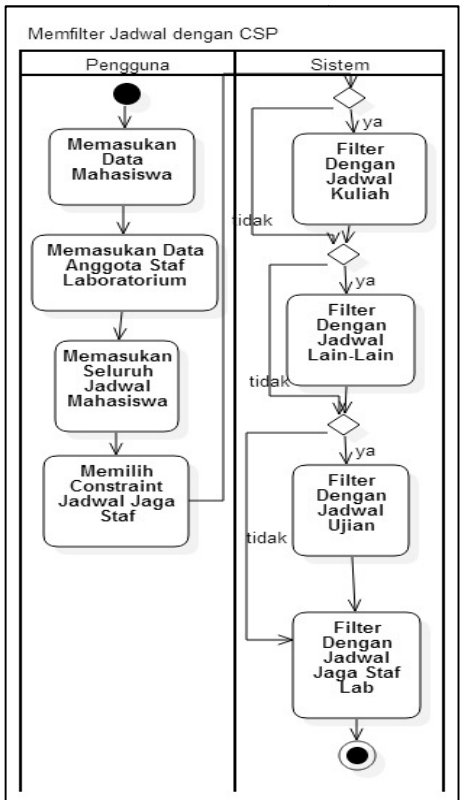
Gambar 1. *Use Case* Pengembangan Aplikasi

B. Analisis Constraint

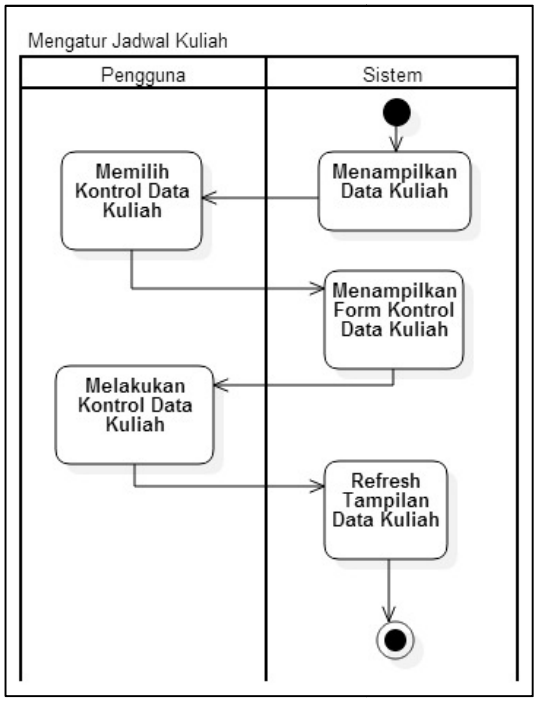
Pada bagian ini akan dijelaskan tentang analisis aplikasi dan penggunaan algoritma dalam pendekatan yang CSP. Pada pendekatan yang diusulkan ini, yang dianggap sebagai variabel adalah jadwal jaga laboratorium, dengan domain adalah para staf yang bertugas dalam jadwal jaga laboratorium.

Hal-hal di bawah ini merupakan persyaratan yang dikelompokkan sebagai Hard *Constraint*:

1. Jadwal kuliah mahasiswa
   Anggota Staf Lab tidak boleh memiliki jadwal jaga staf yang bersamaan dengan jadwal kuliah anggota staf tersebut,
2. Jadwal ujian mahasiswa
   Anggota Staf Lab tidak boleh memiliki jadwal jaga staf yang bersamaan dengan jadwal ujian anggota staf tersebut.
3. Jadwal lain-lain mahasiswa
   Anggota Staf Lab tidak boleh memiliki jadwal jaga staf yang bersamaan dengan jadwal lain-lain anggota staf tersebut. Contoh: Jadwal Asisten Dosen, Jadwal Organisasi Mahasiswa, dll.
4. Seorang staf lab tidak boleh memiliki jadwal jaga yang *overlap* (jaga lebih dari satu *shift* di hari dan jam yang sama).



Gambar 2. *Acivity Diagram* Memfilter Jadwal dengan CSP

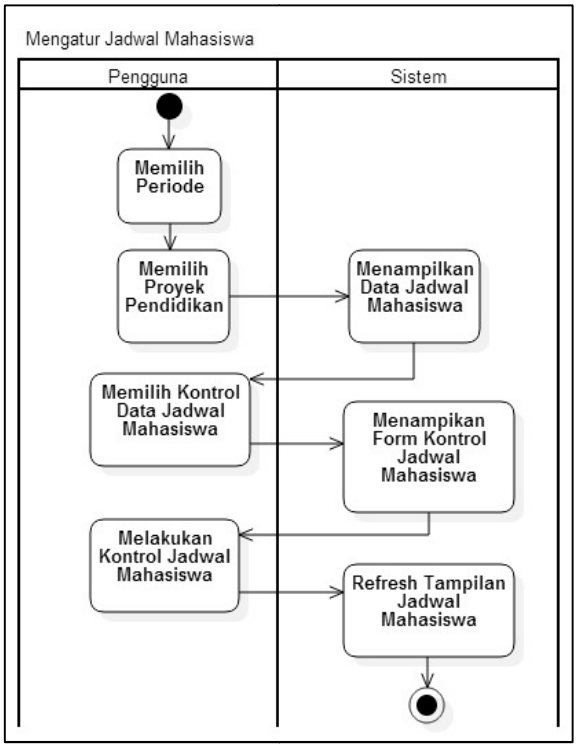


Gambar 3. *Activity Diagram* Mengatur Jadwal Kuliah

Soft Constraint yang digunakan adalah sebagai berikut:

1. Untuk Anggota Staf Lab yang berjenis kelamin wanita diharapkan tidak berjaga di atas pukul 19.00.
2. Jumlah jam pada satu *shift* yaitu 3 jam.
3. Jumlah staf yang berjaga pada 1 *shift* yaitu 2 orang.
4. Jam mulai bekerja pada hari Senin – Jumat yaitu pukul 07.00 – 22.00, sedangkan Sabtu pukul 07.00-19.00.
5. Semua Anggota Staf Lab mendapatkan pembagian jadwal yang sama rata.

Pada *soft constraint* di atas, terdapat suatu prioritas utama, yaitu ketika harus membuat pembagian jadwal yang sama rata untuk semua staf. Dalam pembuatan jadwal yang sama rata, harus diperhatikan juga bahwa untuk jadwal di atas pukul 19.00 lebih baik untuk staf yang berjenis kelamin pria dan juga jadwal-jadwal tersebut tidak boleh bentrok dan *overlap* dengan jadwal kuliah, jadwal lain-lain, dan jadwal ujian.



Gambar 4. *Activity Diagram* Mengatur Jadwal Mahasiswa

C. Analisis untuk Jadwal Bentrok

Pada pembuatan jadwal staf, yang paling perlu diperhatikan adalah jadwal tidak boleh ada yang bentrok dengan jadwal lainnya, sehingga tidak akan ada masalah ketika jadwal dibentuk [8]-[9]. Dalam konteks penelitian ini, dapat ditentukan bahwa jadwal tidak bentrok dengan jadwal lainnya, berdasarkan waktu mulai jadwal dan waktu selesai jadwal.

Contoh: apabila ada Jadwal A dan Jadwal B yang tidak boleh bentrok, maka setiap jadwal harus ada waktu mulai

dan waktu selesai, dan tidak boleh memenuhi kondisi berikut:

1. Waktu mulai jadwal A lebih besar atau sama dengan waktu mulai jadwal B dan waktu selesai jadwal A lebih kecil atau sama dengan waktu selesai jadwal B.
2. Waktu mulai jadwal A lebih besar atau sama dengan waktu mulai jadwal B dan waktu mulai jadwal A lebih kecil dari waktu selesai jadwal B.
3. Waktu selesai jadwal A lebih besar dari waktu mulai jadwal B dan waktu selesai jadwal A lebih kecil atau sama dengan dari waktu selesai jadwal B.
4. Waktu mulai jadwal B lebih besar atau sama dengan waktu mulai jadwal A dan waktu selesai jadwal B lebih kecil sama dengan waktu selesai jadwal A.

Apabila kondisi waktu mulai dan selesai jadwal A dan B tidak ada yang memenuhi kondisi di atas, maka dapat dipastikan bahwa jadwal tersebut tidak bentrok.

IV. IMPLEMENTASI

A. *Implementasi Algoritma CSP*

Pada pembuatan jadwal jaga staf laboratorium, perlu memperhatikan *hard constraint* jadwal kuliah, jadwal mahasiswa, jadwal ujian dan jadwal *overlap*. Untuk menyelesaikan kasus tersebut, maka pada saat pembuatan jadwal jaga, setiap waktu pada jadwal jaga akan dibandingkan dan difilter oleh setiap *hard constraint*, yaitu: pertama dibandingkan dengan Jadwal Kuliah, kemudian dibandingkan lagi oleh Jadwal Mahasiswa, lalu dibandingkan lagi oleh Jadwal Ujian, dan terakhir dibandingkan dengan jadwal staf itu sendiri yang telah dibuat di waktu yang sama supaya tidak ada jadwal yang *overlap* (bentrok).

Untuk mengatasi pembagian jadwal yang sebaiknya sama rata, dan apabila ada perbedaan waktu pembagian, masih dapat ditolerir agar perbedaan tidak terlalu berbeda dan timpang sebelah [10]-[13]. Maka untuk jadwal yang paling banyak *constraint* dan hanya satu staf yang dapat berjaga harus diisi terlebih dahulu. Oleh karena itu, aplikasi akan mengisi terlebih dahulu jadwal staf yang di atas pukul 19.00 dan juga jadwal jaga yang hanya ada 1 staf saja yang dapat berjaga. Hal ini untuk menghindari kasus yang akan dijelaskan pada Gambar 5. Hal yang dijelaskan pada Gambar 5 dapat diatasi dengan pengisian jadwal yang paling banyak memiliki *constraint* terlebih dahulu (*most constrained variable*), dengan tujuan untuk menghasilkan pembagian jadwal untuk semua staf dengan jumlah jam jaga yang sama rata.

B. *Implementasi untuk Menentukan Jadwal Bentrok*

Untuk menentukan jadwal yang bentrok maka jadwal tidak boleh memenuhi kondisi jadwal bentrok. Berdasarkan pada analisis pada bagian sebelumnya, maka dapat dituliskan aturan bertingkat sebagai prioritas penentuan keadaan bentrok jadwal jaga seorang mahasiswa, sebagai berikut:

Contoh	mulai	selesai	
Jadwal Shift	16:00	19:00	
	19:00	22:00	
Staf yang dapat berjaga	Rina	Female	
	Doni	Male	
Apabila Rina tidak dimasukkan pertama ke dalam jadwal, maka akan ada kemungkinan seperti ini	mulai	selesai	
	16:00	19:00	Doni
	19:00	22:00	
			<Kosong/Dianggap 'male' tidak ada yang bisa kecuali Doni berjaga lagi> apabila diisi Doni, maka Rina tidak akan mendapatkan jam jaga
Maka untuk memenuhi kedua Soft Constraint diatas, dilakukan pengisian jadwal mulai dari yang paling banyak constraint, yaitu jadwal mulai pukul 19:00	**Step 1**		
	mulai	selesai	
	16:00	19:00	<kosong/blm terisi>
	19:00	22:00	Doni
	Step 2		
	mulai	selesai	
	16:00	19:00	Rina
	19:00	22:00	Doni
Maka jadwal yang lebih banyak constraint akan diprioritaskan terlebih dahulu untuk diisi agar selain semua jadwal terisi, semua staf juga kebagian jam jaga yang sama rata jumlahnya			

Gambar 5. Masalah Bentrok yang Mungkin Terjadi

```
Waktu mulai jadwal A >= waktu mulai jadwal B
AND waktu selesai jadwal A lebih <= waktu
selesai jadwal B
Waktu mulai jadwal A >= waktu mulai jadwal B
AND waktu mulai jadwal A < waktu selesai
jadwal B
Waktu selesai jadwal A > waktu mulai jadwal
B AND waktu selesai jadwal A <= waktu
selesai jadwal B
Waktu mulai jadwal B >= waktu mulai jadwal A
AND waktu selesai jadwal B <= waktu selesai
jadwal A
```

Dengan demikian, untuk menemukan jadwal kuliah, jadwal mahasiswa, jadwal ujian, dan jadwal yang *overlap* dalam pembuatan jadwal jaga staf, setiap jadwal tidak boleh ada yang memenuhi kondisi jadwal yang bentrok seperti di atas.

Setiap jenis jadwal untuk seorang staf, akan diseleksi satu per satu dengan urutan prioritas sebagai berikut:

1. Seleksi staf untuk jadwal jaga dengan jadwal kuliah staf tersebut.
2. Seleksi staf untuk jadwal jaga dengan jadwal (lain-lain) mahasiswa staf tersebut.
3. Seleksi staf untuk jadwal jaga dengan jadwal ujian sesuai mata kuliah yang diambil staf tersebut.
4. Seleksi staf untuk jadwal jaga dengan jadwal jaga pada waktu yang sama untuk menghindari jadwal yang *overlap*.

Pada seleksi staf yang berjaga berdasarkan jadwal kuliah, jadwal mahasiswa, dan jadwal ujian hanya akan dilakukan apabila pengguna memilih untuk menyusun jadwal berdasarkan jadwal-jadwal tersebut.

Pengguna dapat memilih menyusun jadwal jaga berdasarkan jadwal kuliah, jadwal (lain-lain) mahasiswa, atau jadwal ujian sesuai kebutuhan. Jadwal kuliah, jadwal mahasiswa, dan jadwal ujian dapat dipilih pada *checkbox* yang telah disediakan oleh aplikasi. Untuk menjamin konsistensi terpenuhinya suatu persyaratan, dalam pengembangan dilakukan implementasi konsistensi domain [6]-[8], dengan memperhatikan jadwal bentrok dan prioritas pemenuhan *constraint* sebagaimana telah dituliskan sebelumnya. Prosedur untuk pengecekan konsistensi domain yang digunakan dalam penelitian ini dapat dilihat selengkapnya dalam Algoritma 1.

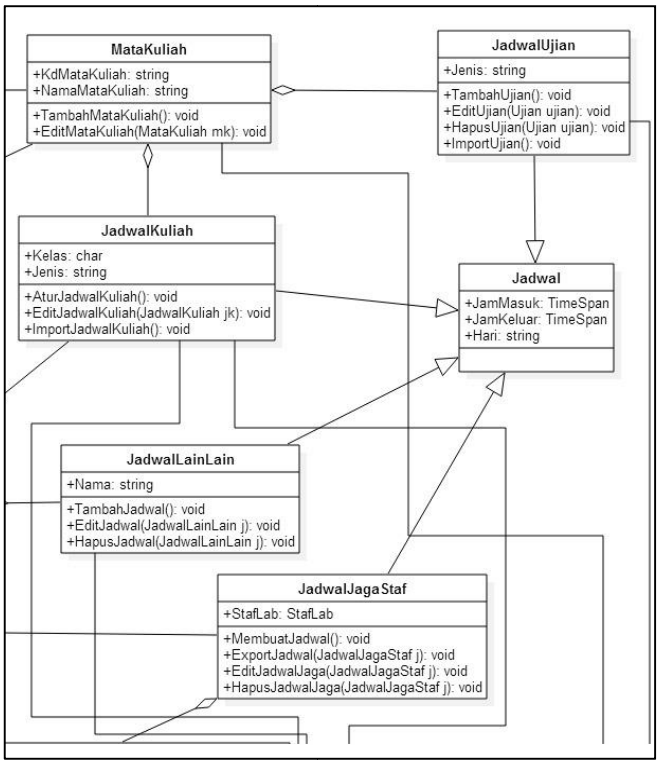
```
1.  Procedure MembuatJadwal(listShift,listStaf, constraintJadwal)
2.  Inputs
3.        listShift        : satu set variabel untuk kasus penjadwalan
4.        listStaf         : satu set domain untuk kasus penjadwalan
          constraintJadwal : constraint yang berisi nilai constraint, apakah jadwal  akan di
          filter oleh JadwalKuliah, JadwalLain-lain, atau JadwalUjian
5.  Output
6.       listShift (satu set variabel yang telah diisikan oleh domain)
7.  Process
8.       tempListStaf      : satu set list untuk menampung sementara staf hasil filter
9.       tmpStaf           : satu variabel untuk menampung sementara secara staf
10.      For each shift in listShift do
11.           tempListStaf = filterStaf(listStaf,shift,constraintJadwal)
12.           If(tempListStaf.Count <= 2 && tempListStaf.Count > 0) then
13.                Shift.Staf = tempListStaf[0]
14.           Else If(shift.jamMulai >= 19.00) then
15.                tempListStaf = From item in tempListStaf
16.                               Where item.gender == 'Male'
17.                               Select item
18.                tmpStaf =  From item in tempListStaf
19.                               Having Min(item.totalJaga() )
20.                               Select item
21.                If(tmpStaf != null) then
22.                      Shift.Staf = tmpStaf
23.                Endif
24.           Endif
25.      End For each
26.      For each shift in listShift do
27.         tempListStaf = filterStaf(listStaf,shift,constraintJadwal)
28.         While not cekBentrok (listStaf, shift)
29.                tempListStaf = From item in tempListStaf
30.                       Where item.totalJaga() == Min(item.TotalJaga())
31.                       Select item
32.                tmpStaf = selectRandomFrom(tmpStaf)
33.                If(tmpStaf != null) then
34.                       Shift.Staf = tmpStaf
35.                Endif
36.           End While
37.       End For Each
38. Return listShift
```

Algoritma 1. Prosedur Pengecekan Konsistensi Domain dengan Prioritas (*method* MembuatJadwal())

Dalam Algoritma 1, konsistensi pengisian variabel untuk *shift* jadwal jaga dengan domain ketersediaan staf mahasiswa, dimulai dengan memilih *constraint* sebagai batasan terbesar (*most constrainted variable*), yaitu jadwal jaga di atas jam 19.00 (baris 15-20). Namun karena itu adalah sebuah soft-*constraint*, maka jika tidak terisi, masih dapat dilanjutkan dengan pengisian *hard-constraint*-nya (baris 26-37). Pengecekan kondisi prioritas bentrok dilakukan setlah didapatkan kandidat staf yang mengisi *shift* tertentu (baris 28). Untuk menjamin bahwa tidak ada staf yang mengisikan *shift* terlalu banyak, maka setiap kali pemilihan staf dilakukan, perlu dipilih staf dengan jumlah *shift* minimal (baris 29-32). Hal ini dilakukan sampai semua *shift* dapat diisi dengan staf mahasiswa yang tersedia.

C. Implementasi Program Aplikasi

Implementasi keseluruhan sistem dilakukan dalam bahasa pemrograman C#, melalui antar muka windows *form*, berbentuk aplikasi desktop. Pada pembuatan aplikasi, digunakan Visual Studio 2013 dengan .Net *Framework* 4.5 [14], dengan memanfaatkan sistem basis data SQL Server 2012 [15]. Cuplikan *class diagram* untuk sistem pembangkitan jadwal yang dikembangkan dapat dilihat dalam Gambar 6.



Gambar 6. Cuplikan *Class Diagram* Implementasi Pembangkitan Jadwal

Ditinjau dari sisi pengembangan perangkat lunak, pembangkitan jadwal dilakukan dengan memanfaatkan implementasi berbasis pola perangkat lunak *command* / perintah. Dalam pola ini sebuah permohonan pembangkitan jadwal Staf laboratorium dilakukan sebagai sebuah obyek (lihat properti 'StafLab' dalam kelas 'JadwalJagaStaf', yang diinstasiasikan sebagai obyek Staflab. Dengan cara seperti ini maka pembangkitan jadwal dapat dilakukan secara bertahap dan setiap kali sudah terbentuk kandidat jadwal, dapat langsung dilihat oleh koordinator staf untuk diverifikasi, tanpa harus menunggu semua kandidat tersedia [16, 17].

Terkait dengan implementasi *constraint* dalam bentuk jadwal setiap mahasiswa, implementasi dilakukan dengan konsep pola '*decorator*', sehingga setiap jadwal adalah

sebuah kelas khusus yang dapat dikombinasikan dengan jadwal lainnya [16, 17]. Jadwal dasar yang harus ada adalah jadwal mata kuliah untuk setiap staf, sedangkan jadwal lainnya dapat ditambahkan secara fleksibel.

V. EVALUASI DAN PEMBAHASAN HASIL

A. *Evaluasi*

Dalam bagian ini disampaikan evaluasi yang dilakukan untuk menilai sejauh mana jadwal yang berhasil dibangkitkan bermanfaat dalam realitas. Evaluasi dilakukan dengan cara:

1. Membandingkan hasil pembangkitan jadwal dengan yang sebelumnya telah dibuat oleh koordinator staf secara manual. Untuk hal ini menggunakan data dari penjadwalan staf laboratorium FIT semester ganjil 2015/16.
2. Melakukan survei pemanfaatan melalui wawancara dengan koordinator dan staf laboratorium FIT lainnya.

Contoh hasil pembangkitan jadwal dapat dilihat dalam Gambar 7 dan 8. Gambar 7 memberikan jadwal akhir yang siap untuk dicetak dan / atau di-*export* ke dalam bentuk *file* .xlsx. Gambar 8 (setelah daftar pustaka), memberikan contoh salah satu kandidat jadwal yang dihasilkan.

Setiap kolom dalam Gambar 8 memberikan contoh jadwal untuk setiap staf yang berhasil dibentuk, dan dibandingkan dengan yang disusun secara manual. *Constraint* yang digunakan untuk membatasi adalah jadwal mata kuliah dan jadwal lain-lain. Hasil umum yang dapat dilihat adalah:

1. Jadwal yang dihasilkan memenuhi syarat *soft constraint*, yaitu staf yang berjaga di atas pukul 19.00 berjenis kelamin laki-laki.
2. Jadwal yang dihasilkan memiliki pembagian jam jaga untuk seluruh anggota staf dengan jumlah yang sama rata.
3. Jadwal yang dihasilkan oleh program lebih akurat, dibandingkan dengan jadwal versi awal yang dibuat oleh koordinator secara manual. Hal tersebut dapat dilihat pada jadwal hari Senin pukul 16.00, terdapat sebuah jadwal yang bentrok dengan mata kuliah.

Jadwal Staf Lab FIT
Jadwal Reguler 1

Monday	07:00 - 10:00	1373006	Chrysanthia Novelia
Monday	07:00 - 10:00	1372062	Benedict
Monday	10:00 - 13:00	1373022	Frans Mulia Hidayat
Monday	10:00 - 13:00	1373006	Chrysanthia Novelia
Monday	10:00 - 13:00	1472027	Yorenvo Stevano
Monday	13:00 - 16:00	1373006	Chrysanthia Novelia
Monday	13:00 - 16:00	1373011	Billyanto Hendrik
Monday	13:00 - 16:00	1372004	Yosef Oktavianus Gunawan
Monday	16:00 - 19:00	1472013	Jason Alexander
Monday	16:00 - 19:00	1373016	Jessica Christie
Monday	19:00 - 22:00	1372098	William
Monday	19:00 - 22:00	1473009	Elbert Kelvin Nugroho
Tuesday	07:00 - 09:30	1372071	Willy Kurniawan
Tuesday	07:00 - 09:30	1372011	Chandra Ari Gunawan
Tuesday	09:30 - 13:00	1472027	Yorenvo Stevano
Tuesday	09:30 - 13:00	1472008	Chandra Vincent Grafianto
Tuesday	13:00 - 16:00	1372007	Imelda Tirta Saputra
Tuesday	13:00 - 16:00	1472002	Elvina
Tuesday	13:00 - 16:00	1472085	Yonatan Abiezer
Tuesday	16:00 - 19:00	1372071	Willy Kurniawan
Tuesday	16:00 - 19:00	1372007	Imelda Tirta Saputra
Tuesday	19:00 - 22:00	1473009	Elbert Kelvin Nugroho
Tuesday	19:00 - 22:00	1372019	Sam Chandraka
Wednesday	07:00 - 10:00	1472085	Yonatan Abiezer
Wednesday	07:00 - 10:00	1472006	Sofriesillero Zumaydis
Wednesday	10:00 - 13:00	1373016	Jessica Christie
Wednesday	10:00 - 13:00	1472086	Bertha Alan Manuel
Wednesday	10:00 - 13:00	1472008	Chandra Vincent Grafianto
Wednesday	13:00 - 16:00	1472085	Yonatan Abiezer
Wednesday	13:00 - 16:00	1372020	Gabriela Stefiyani Suprato
Wednesday	13:00 - 16:00	1372062	Benedict
Wednesday	16:00 - 19:00	1472019	Jeson Orlando Isakh
Wednesday	16:00 - 19:00	1372020	Gabriela Stefiyani Suprato
Wednesday	19:00 - 22:00	1472019	Jeson Orlando Isakh
Wednesday	19:00 - 22:00	1473009	Elbert Kelvin Nugroho
Thursday	07:00 - 10:00	1372004	Yosef Oktavianus Gunawan
Thursday	07:00 - 10:00	1472008	Chandra Vincent Grafianto
Thursday	10:00 - 13:00	1372098	William
Thursday	10:00 - 13:00	1372019	Sam Chandraka
Thursday	10:00 - 13:00	1372020	Gabriela Stefiyani Suprato
Thursday	13:00 - 16:00	1372004	Yosef Oktavianus Gunawan
Thursday	13:00 - 16:00	1372008	Felix Christian Jonathan
Thursday	13:00 - 16:00	1472013	Jason Alexander
Thursday	16:00 - 19:00	1373022	Frans Mulia Hidayat
Thursday	16:00 - 19:00	1472002	Elvina
Thursday	19:00 - 22:00	1472088	Steven Yong
Thursday	19:00 - 22:00	1472027	Yorenvo Stevano
Friday	07:00 - 10:00	1472088	Steven Yong
Friday	07:00 - 10:00	1472086	Bertha Alan Manuel
Friday	10:00 - 13:00	1372007	Imelda Tirta Saputra
Friday	10:00 - 13:00	1472006	Sofriesillero Zumaydis
Friday	13:00 - 16:00	1372011	Chandra Ari Gunawan
Friday	13:00 - 16:00	1472013	Jason Alexander
Friday	16:00 - 19:00	1372071	Willy Kurniawan
Friday	16:00 - 19:00	1373011	Billyanto Hendrik
Friday	19:00 - 22:00	1373011	Billyanto Hendrik
Friday	19:00 - 22:00	1372011	Chandra Ari Gunawan
Saturday	07:00 - 10:00	1372008	Felix Christian Jonathan
Saturday	07:00 - 10:00	1372098	William
Saturday	10:00 - 13:00	1472019	Jeson Orlando Isakh
Saturday	10:00 - 13:00	1372019	Sam Chandraka
Saturday	13:00 - 16:00	1472086	Bertha Alan Manuel
Saturday	13:00 - 16:00	1472085	Yonatan Abiezer
Saturday	16:00 - 19:00	1472002	Elvina
Saturday	16:00 - 19:00	1472088	Steven Yong

Gambar 7. Jadwal Akhir untuk Dicetak dan / atau Di-*export* dalam Format *.xlsx

Terlihat bahwa penerapan algoritma CSP berhasil memberikan usulan jadwal yang beririsan dengan yang disusun manual sebesar 16/23, sekitar 70% (arsiran berwarna biru). Lebih jauh, algoritma CSP berhasil memberikan usulan jadwal untuk semua staf (warna jingga). Jika dibandingkan dengan jadwal yang dibuat secara manual, maka hasil dari algoritma CSP meskipun berbeda slot waktu, namun tetap memenuhi *constraint* yang membatasi kesediaan setiap staf. Jadi meskipun tidak beririsan dengan jadwal yang dibuat secara manual, namun memberikan kandidat yang dapat digunakan untuk penjadwalan akhir. Hasil pada Gambar 8 tersebut menunjukkan suatu potensi bahwa jadwal yang berhasil dibangkitkan melalui penerapan CSP memiliki kualitas yang setara dengan yang dihasilkan secara manual.

Hasil wawancara dengan koordinator staf menunjukkan bahwa penerapan algoritma CSP untuk pembangkitan jadwal telah memberikan hasil yang baik, dan akan sangat menolong dalam memberikan kandidat penjadwalan, serta tidak perlu lagi memakan waktu yang lama. Dalam kasus uji coba, dengan data staf laboratorium FIT di Universitas XXX pada semester ganjil 2015/16, aplikasi memerlukan waktu rata-rata 3,5 menit untuk membuat sebuah kandidat penjadwalan, dan ini sangat berbeda jauh dengan yang biasa dilakukan secara manual, yaitu sekitar 1 hari kerja untuk menghasilkan satu kandidat penjadwalan.

Beberapa masukan yang berhasil dihimpun dari hasil wawancara dengan 3 koordinator staf dan 20 staf, adalah sebagai berikut:

1. Dapat menambahkan fitur pengiriman notifikasi email secara otomatis kepada seluruh staf, untuk menginformasikan jadwal akhir yang diputuskan oleh koordinator staf, dan mengingatkan waktu jaga laboratorium.
2. Menyederhanakan tampilan (*form*) sehingga dapat lebih intuitif dalam hal penggunaan. Tampilan utama dari aplikasi yang dikembangkan selama penelitian berlangsung dapat dilihat dalam Gambar 9 (setelah daftar pustaka).

B. Pembahasan Hasil

Mengacu pada hasil implementasi dan uji coba sebagaimana diuraikan dalam bagian sebelumnya, maka hasil-hasil tercapai tersebut dapat dikatakan memberikan kontribusi ilmiah dalam penelitian penjadwalan, sebagai berikut:

1. Implementasi Algoritma 1, disertai dengan penanganan bentrok jadwal telah mampu memberikan prioritas yang mengikuti preferensi staf laboratorium dan kebutuhan penjadwalan itu sendiri. Melalui penanganan tersebut, aplikasi memberikan kontribusi yang disarankan dalam penelitian Lapègue, dkk [11] untuk selain memperhatikan pembagian jadwal secara merata juga memperhitungkan preferensi staf.
2. Implementasi Algoritma 1 secara berprioritas juga memberikan alternatif untuk menghindari penggunaan teknik optimasi, sehingga mampu mengatasi permasalahan pemilihan fungsi obyektif yang disampaikan dalam penelitian [10] dan [13].
3. Implementasi Algoritma 1 beserta penanganan prioritas dan pemerataan jam kerja di dalam sistem penjadwalan staf laboratorium juga telah memberikan alternatif dalam pembangkitan jadwal untuk jumlah staf yang tidak terlalu banyak, antara 20-30 orang. Dengan adainya alternatif-alternatif yang dapat dipilih dan dikombinasikan dengan prioritas yang fleksibel, maka sistem lengkap yang diusulkan melalui makalah dapat mengatasi permasalahan verifikasi dan fleksibilitas penjadwalan sebagaimana diangkat dalam penelitian [12].

VI. SIMPULAN DAN SARAN

A. Simpulan

Dari hasil pengujian yang telah dilakukan dengan jadwal yang telah ada dan juga berdasarkan survei yang dilakukan pada kepala laboratorium (sebagai penanggung jawab staf laboratorium FIT) dan Koordinator utama staf (sebagai pembuat jadwal), dapat ditarik beberapa kesimpulan, yaitu:

1. Jadwal yang dihasilkan sudah cukup baik dan dapat digunakan untuk jadwal jaga staf laboratorium.
2. Fitur *import* dan *export* data sangat berguna dalam penggunaan aplikasi terutama untuk memudahkan masukan jadwal mata kuliah yang berasal dari sistem akademik.
3. Aplikasi yang telah dikembangkan dirasakan dapat berkontribusi dalam pembuatan jadwal jaga Staf Laboratorium FIT di Universitas XXX.

B. Saran

Berdasarkan hal-hal yang telah dicapai dalam implementasi, diperoleh beberapa saran dari responden, yaitu:

1. Memperbaiki implementasi kelas-kelas dalam aplikasi sehingga memiliki kemampuan untuk generalisasi pembentukan *constraint* dan/atau pembangkitan jadwal pada kasus lain di luar administrasi staf laboratorium, namun dengan konteks persyaratan yang mirip.
2. Desain antarmuka aplikasi yang dibuat lebih sederhana, konsisten, dan menarik sehingga memudahkan pengguna untuk memahami penggunaan aplikasi.
3. Untuk meningkatkan tingkat pengoperasian aplikasi secara mandiri, di masa mendatang, aplikasi diharapkan dapat langsung terhubung langsung dengan sistem akademik.

DAFTAR PUSTAKA

[1] Tsang, Edward. Foundations of Constraint Satisfaction: The Classic Text. BoD–Books on Demand, 2014.

[2] Baptiste, Philippe, Claude Le Pape, and Wim Nuijten. Constraint-based scheduling: applying constraint programming to scheduling problems. Vol. 39. Springer Science & Business Media, 2012.

[3] Laborie, Philippe. "Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results." Sixth European Conference on Planning. 2014.

[4] Van den Bergh, Jorne, et al. "Personnel scheduling: A literature review."European Journal of Operational Research 226.3: 367-385. 2013.

[5] Pinedo, Michael L. Scheduling: theory, algorithms, and systems. Springer Science & Business Media, 2012.

[6] Russel, Stuart J., Norvig, Peter. Prentice Hall. Artificial Intelligence: A Modern Approach (3rd ed). 2009.

[7] Luger, George F. Addison Wesley. Artificial Intelligence: Structures and strategies for Complex Problem Solving. 6th Edition. 2008.

[8] Poole, David L., Alan K. Mackworth. Artificial Intelligence: Foundations of Computational Agents. Cambridge University Press. 2010.

[9] Watson, Mark. Practical Artificial Intelligence Programming in Java. Lean Publisher. 2013.

[10] Hadwan M, Ayob M, Sabar NR, Qu R. A harmony search algorithm for nurse rostering problems. Information Sciences. 1;233:126-40. Jun. 2013.

[11] Lapègue T, Bellenguez-Morineau O, Prot D. A constraint-based approach for the shift design personnel task scheduling problem with equity. Computers & Operations Research. 31;40(10):2450-65. Okt. 2013.

[12] Ásgeirsson, Eyjólfur Ingi. "Bridging the gap between self-schedules and feasible schedules in staff scheduling." Annals of Operations Research 218.1: 51-69. 2014.

[13] Meskens, Nadine, David Duvivier, and Arnauld Hanset. "Multi-objective operating room scheduling considering desiderata of the surgical team." Decision Support Systems 55.2: 650-659. 2013.

[14] Microsoft Corporation. (n.d.). Introduction to the C# Language and the .NET Framework. (Microsoft). [Online] Akses: Mei 18, 2015. Tersedia: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

[15] Microsoft Corporation. (n.d.). Sql Server 2012. [Online]. Akses: Mei 18, 2015. Tersedia https://msdn.microsoft.com/en-us/library/ff929050(v=sql.10).aspx

[16] Bruegge, Bernd, and Allen H. Dutoit. Object-Oriented Software Engineering Using UML, Patterns and Java 3th ed.. Pearson, 2010.

[17] Kerievsky, Joshua. Refactoring to patterns. Pearson Deutschland GmbH, 2005/

Jadwal Reguler		Jeson	Yorenvo	S.Yong	Chandra	Sam	William	Bertha	Chrysan	Billyanto	C.Vincent	Yonathan	E.Kelvin	Jason A	Willy	Elvina	Stefi	Benedict	Yosef	Felix	Frans	Sofrie	Jessica	Imelda
Senin	07.00-10.00			Asd 2	Menpro	Menpro		PBOA 1			Asd 2	PBOA 1	PSI	Asd 2	Menpro	PBOA 1	Menpro		Menpro	Menpro	PSI	PBOA 1		Menpro
	10.00-13.00	Basdat 1		Asd 2	Menpro	Menpro		PBOA 1			Asd 2	PBOA 1		Asd 2	Menpro	PBOA 1	Menpro		Menpro	Menpro		PBOA 1	EH 1	Menpro
	13.00-16.00	Basdat 1		RPL1		Asdos		Probis			RPL1	RPL1	SI SDM		Tekom		Stralgo	Tekom				RPL1	EH 1	Asdos
	16.00-19.00	Basdat 1		RPL1		Asdos					RPL1	RPL1	SI SDM		Tekom		Stralgo	Tekom				RPL1		Asdos
	19.00-22.00																							
Selasa	07.00-10.00	Matdis	Matdis	Tekmul 1	09:30	PDPL	TemuPen	Stralgo	Pem. SI	Pem. SI	Matdis	Matdis	Jarkom	Tekmul 1	09:30	Stralgo	TemuPen	Stralgo	PDPL	Probis	Pemodel	Stralgo	PSI	TemuPen
	10.00-13.00	Tekmul 1	09:30	Matdis	Probis	Admin B	TemuPen	Basdat1	PWL	PWL	09:30	Tekmul1	Jarkom	Basdat1	Probis	Matdis	TemuPen	Probis	Admin B	Asdos	Pemasar	Tekmul 1	PWL	TemuPen
	13.00-16.00						AdminB	Basdat1		PWL				Basdat1			Probis		Probis	Asdos	PWL	Probis		
	16.00-19.00	B Ind	B Ind	B Ind			AdminB	Basdat1	UID	UID	B Ind	PAK		Basdat1		B Ind				Asdos	UID		UID	
	19.00-22.00	B Ind	B Ind	B Ind					UID	UID	B Ind	PAK				B Ind					UID		UID	
Rabu	07.00-10.00	Kepem		Kepem	Machine	PDPL	Menpro	Kepem	PWL	Kepem	Kepem		Strukdat	Kepem	Kepem	Asdos	Progweb	Menpro	PDPL	Machine	Kepem		PWL	
	10.00-13.00	Asd 2	Asd 2	Kepem	Machine	Asdos	Progweb2			PWL		Asd 2	Strukdat	Kepem	Kepem	Comptive	Progweb	Kepem	Progweb	Machine	PWL	Comptive		Progweb
	13.00-16.00	Asd 2		Asdos	Comptive	Stralgo	Progweb2	Tekmul1	Menpro	Menpro	Tekmul1		Fenom	Matdis	Stralgo	Comptive			Progweb	Progweb2	Menpro	Comptive	Menpro	Progweb
	16.00-19.00		Tekmul 1	PAK	Comptive	Progweb2	Progweb2	Tekmul1	Menpro	Menpro	Tekmul1	B Ind		PAK	B ind	Comptive		PWE	PWE	Progweb2	Menpro	Comptive	Menpro	Progweb
	19.00-22.00		Tekmul 1	PAK	Progweb1	Progweb2						B Ind		PAK	B ind	Tekmul 1		PWE	PWE			PAK		
Kamis	07.00-10.00	RPL 1	RPL 1		PDPL	Asdos	PDPL							RPL1		RPL 1				PDPL				PDPL
	10.00-13.00	RPL 1	Basdat1	Basdat1	PDPL						Basdat1	Basdat1	BDL	RPL1		RPL 1				PDPL	Rekayasa	Basdat1		PDPL
	13.00-16.00		Basdat1	Basdat1	Asdos		Stralgo	RPL 1	Kepem	PSI	Basdat1	Basdat1	BDL								Rekayasa	Basdat1	Kepem	Stralgo
	16.00-19.00		Basdat1	Basdat1	Progweb1	Progweb2		RPL 1	PSI	PSI	Basdat1	Basdat1	Etika	B Ind				B Ind	Asdos	Asdos		Basdat1	PSI	
	19.00-22.00				Progweb1	Progweb2			PSI	PSI			Etika	B Ind				B Ind	Asdos	Asdos			PSI	
Jumat	07.00-10.00		ASD 2		KP	KP	KP						PSI				KP		KP	KP	PSI			KP
	10.00-13.00								EH1						EH 1					Asdos				
	13.00-16.00								EH1						EH 1					Asdos				
	16.00-19.00	PAK	PAK					PAK			PAK					PAK				Asdos				
	19.00-22.00	PAK	PAK					PAK			PAK					PAK				Asdos				
Sabtu	07.00-10.00																							
	10.00-13.00																							
	13.00-16.00																							
	16.00-19.00																							

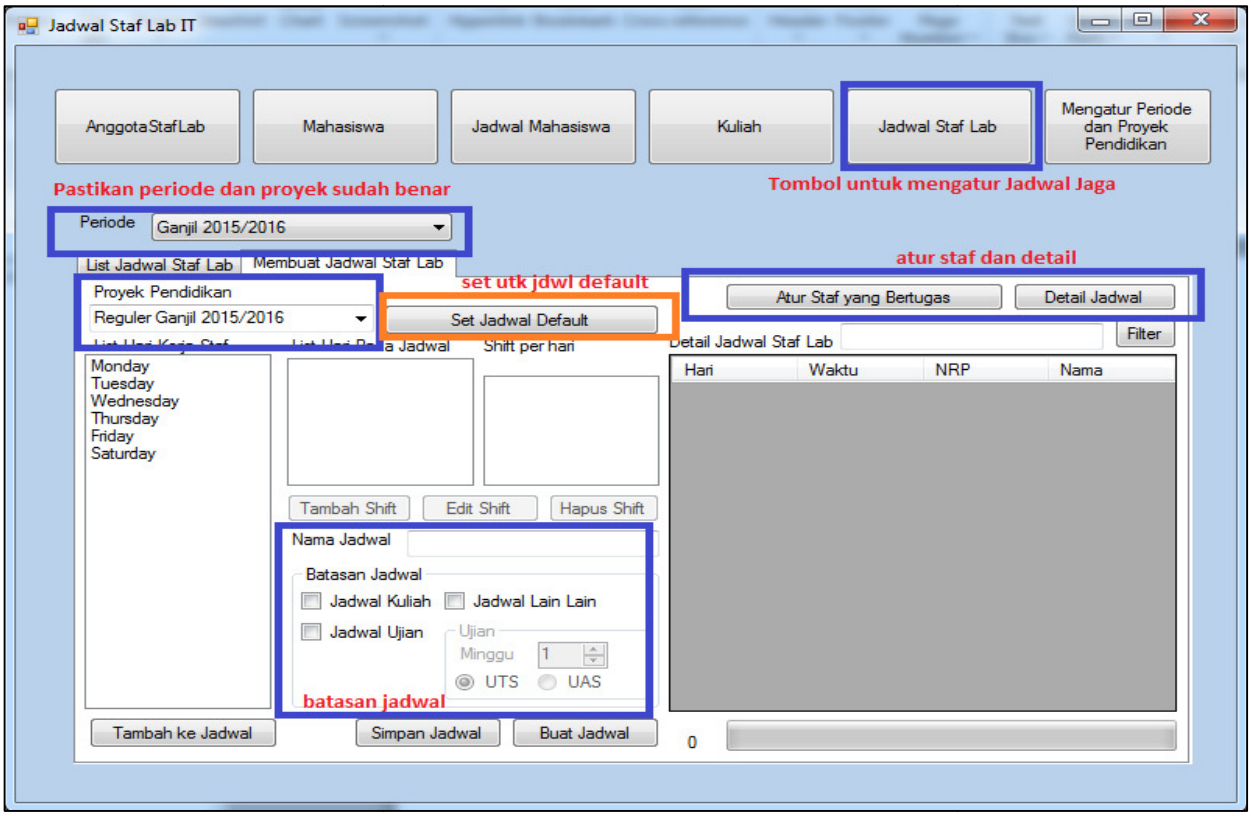
Keterangan Warna :

Jadwal Shift yang dibuat oleh Koordinator Staf Lab Periode Ganjil 2015/2016

Jadwal Shift yang dibuat oleh SistemPenjadwalan dengan CSP Algorithm

Jadwal Shift yang dibuat Koordinator sama dengan jadwal Shift yang dibuat Sistem

Gambar 8. Contoh Hasil Pembangkitan Jadwal dengan CSP Berprioritas



Gambar 9. Tampilan Utama Aplikasi yang Dikembangkan