TUGAS AKHIR – KS141501

OPTIMASI PENJADWALAN MATA KULIAH OTOMATIS DENGAN MENGGUNAKAN ALGORITMA TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC

AUTOMATED COURSE TIMETABLING OPTIMIZATION USING TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC ALGORITHM

REDIAN GALIH IRIANTI
NRP 05211440000036

Dosen Pembimbing
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – KS141501

OPTIMASI PENJADWALAN MATA KULIAH OTOMATIS DENGAN MENGGUNAKAN ALGORITMA TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC

REDIAN GALIH IRIANTI
NRP 05211440000036

Dosen Pembimbing
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

DEPARTEMEN SISTEM INFORMASI
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – KS141501

AUTOMATEDCOURSE TIMETABLING OPTIMIZATION USING TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC ALGORITHM

REDIAN GALIH IRIANTI
NRP 05211440000036

Supervisors
Ahmad Muklason, S.Kom., M.Sc., Ph.D.

INFORMATION SYSTEMS DEPARTMENT

Information and Communication Technology Faculty

Sepuluh Nopember Institut of Technology

Surabaya 2018

LEMBAR PENGESAHAN

OPTIMASI PENJADWALAN MATA KULIAH OTOMATIS DENGAN MENGGUNAKAN ALGORITMA TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh:

REDIAN GALIH IRIANTI
NRP. 05211440000036

Surabaya, 18 Juli 2018

KEPALA
DEPARTEMEN SISTEM INFORMASI

Dr. Ir. Aris Tjahyanto, M.Kom.
NIP 19650310 199102 1 001

LEMBAR PERSETUJUAN

OPTIMASI PENJADWALAN MATA KULIAH OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC*

TUGAS AKHIR

Disusun Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Departemen Sistem Informasi
Fakultas Teknologi Informasi dan Komunikasi
Institut Teknologi Sepuluh Nopember

Oleh :

REDIAN GALIH IRIANTI
NRP. 05211440000036

Disetujui Tim Penguji : Tanggal Ujian : Juli 2018
Periode Wisuda : September2018

Ahmad Muklason, S.Kom., M.Sc., Ph.D. **(Pembimbing I)**

Edwin Riksakomara, S.Kom., MT **(Penguji I)**

Faizal Mahananto, S.Kom, M.Eng., Ph.D. **(Penguji II)**

OPTIMASI PENJADWALAN MATA KULIAH OTOMATIS DENGAN MENGGUNAKAN ALGORITMA *TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC*

Nama Mahasisiwa : Redian Galih Irianti

NRP : 05211440000036

Departemen : Sistem Informasi

Dosen Pembimbing : Ahmad Muklason, S.Kom., M.Sc., Ph.D

ABSTRAK

Rutinitas penjadwalan mata kuliah merupakan salah satu kewajiban bagi setiap perguruan tinggi setiap awal semester baru. Jadwal mata kuliah yang dibuat harus diperbarui dan disesuaikan dengan semester yang sedang berlangsung. Permasalahan terkait dengan penjadwalan mata kuliah masih menjadi topik yang menarik untuk diselesaikan. Banyaknya batasan seperti hard constraint maupun soft constraint harus diperhatikan dalam pembuatan jadwal mata kuliah. Masalah penjadwalan sendiri diklasifikasikan sebagai NP-hard yang berarti permasalahan penjadwalan mata kuliah tidak dapat diselesaikan dengan mudah menggunakan metode konvensional. Penyelesaian masalah penjadwalan mata kuliah erat kaitannya dengan optimasi. Berbagai pendekatan dan metode telah dilakukan dalam penyusunan jadwal mata kuliah untuk mendapat hasil penjadwalan mata kuliah yang optimal dengan melakukan optimasi. Dengan melakukan otomatisasi penjadwalan, diharapkan dapat membantu untuk menghemat waktu penyusunan. Selain hal tersebut, hasil penjadwalan

otomatis juga diharapkan dapat memberikan kenyamanan mahasiswa untuk melaksanakan kuliah hingga berdampak pula pada pencapaian nilai mahasiswa yang maksimal. Tugas akhir ini menggunakan pendekatan hyper-heuristic dengan menggabungkan algoritmaVariable Neighborhood Search dan Tabu Search untuk menyelesaikan permasalahan optimasi penjadwalan mata kuliah. Tugas akhir ini diharapkan dapat membantu perguruan tinggi dalam melakukan salah satu kewajibannya yaitu penjadwalan mata kuliah menjadi lebih optimal serta memberikan kenyamanan bagi mahasiswa. Berdasarkan hasil penelitian tugas akhir ini menunjukkan Algoritma yang digunakan dapat mengoptimalisasi jadwal manual dengan menurunkan nilai pinalti pada semester ganjil sebesar 1948 pinalti dan semester genap sebesar 1281 pinalti pada hasil optimasi.

Kata kunci : penjadwalan mata kuliah, algoritma variable neighborhood search, algoritma tabu search, algoritma hyper-heuristic, hard contraint, soft contraint

AUTOMATED COURSE TIMETABLING OPTIMIZATION USING TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC ALGORITHM

Name	**:**	**Redian Galih Irianti**
NRP	**:**	**05211440000036**
Departement	**:**	**Information Systems**
Supervisior	**:**	**Ahmad Muklason, S.Kom M.Sc., Ph.D**

ABSTRACT

The timetabling course routine is one of the obligations for every college at the beginning of the new semester. Course schedules are made to be updated and adjusted to the current semester. Issues related to course timetabling are still an interesting topic to be resolved. The number of constraints such as hard constraint or soft constraint must be considered in the preparation of the course schedule. Scheduling problems are classified as NP-hard which means course timetabling problems can not be solved easily using conventional methods. Course timetabling problem solving is closely related to optimization. Various approaches and methods have been made in the preparation of the course schedule to obtain optimal course timetabling results by optimizing. By scheduling automation, it is expected to help to save the time of preparation. In addition, the automatic scheduling results are also expected to provide students comfort to carry out the lecture to also impact on the achievement of maximum student value. This final project uses

hyper-heuristic approach by combining algorithms namely Variable Neighborhood Search and Tabu Search to solve the course timetabling optimization problem. From the results of this final project is expected to help the college in performing one of the obligations of scheduling subjects to be more optimal and provide comfort for students. Based on the results of this final project research shows the algorithm that can be used to optimize the manual schedule by lowering the value in the odd semester of 1948 penalties and the even semester of 1281 penalties on the optimization results.

Keywords: course timetabling, variable neighborhood search algorithm, tabu search algorithm, hyper-heuristic algorithm, hard contraint, soft contraint

KATA PENGANTAR

Bismillahirohmanirrohim

Puji Syukur Kepada Allah SWTatas limpahan rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan buku tugas akhir dengan judul

"**OPTIMASI PENJADWALAN MATA KULIAH OTOMATIS DENGAN MENGGUNAKAN ALGORITMA TABU-VARIABLE NEIGHBORHOOD SEARCH BASED HYPER HEURISTIC**"

yang merupakan satu syarat kelulusan pada Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya.
Selama proses awal perkuliahan hingga pengerjaan Tugas Akhir ini, penulis telah memperoleh banyak bantuan, bimbingan dan petunjuk dari berbagai pihak. Oleh karena itu, dalam kesempatan ini penulis akan menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

- Allah SWT, yang telah memberikan kesehatan, kemudahan, kelancaran, keberuntungan dan kesempatan untuk penulissehingga dapat menyelesaikan Tugas Akhir ini.
- Kelurga penulis, bapak Dwi Irianto, Ibu Sudarwati, Mas Tio, Mbak Della dan Dek Nisa yang tidak pernah lelah untuk selalu mendokan, memberikan semangat,kasih sayang dan mendampingi dari awal perkuliahan hingga saat ini dan kedepannya
- Bapak Dr. Ir. Aris Tjahyanto, M.Kom, selaku Ketua Departemen Sistem Informasi ITS, yang telah membantu

menyediakan fasilitas terbaik selama perkuliahan hingga saat ini

- Bapak Ahmad Muklason, S.Kom., M.Sc, Ph.D.selaku Dosen Pembimbing Tugas Akhir yang telah banyak meluangkan waktu dan juga tenaga untuk membimbing, mengarahkan dan mendukung dalam penyelesaian Tugas Akhir ini dengan sabar
- Bapak Prof.Ir. Arif Djuanaidy, M.Sc, Ph.D. selaku dosen wali yang telah memberikan arahan dan wejangan terkait perkuliahan di Departemen Sistem Informasi
- Seluruh dosen pengajar beserta staf dan karyawan di Departemen Sistem Informasi, FTIK ITS Surabaya yang telah memberikan ilmu dan bantuan kepada penulis selama masa perkuliahan
- Sahabat-sahabat dekat penulis Joni, Gusti, Icak, Fufu, Ayusha, Fanny, Risha, Nom, Rosi, Gea, Bram, Alim, Calvin, Berli, Pras yang selalu siap sedia menemani saya selama perkuliahan ini
- Untuk Brilianto Wilis Satria Nugraha yang selalu menyemangati, membantu dan menemani penulis dari awal perkuliahan hingga saat ini dan kedepannya
- Untuk Dewangga dan Adrian yang dengan sabar membantu dan membinbing penulis dalam menyelesaikan Tugas Akhir ini
- Teman-teman satu pembimbing Andy, Gusti, Imad, Silfia, Ujikdan anggota RDIB lainnya yang menemani penulis dan menjadi teman diskusi dalam pengerjaan Tugas Akhir
- Teman-teman kabinet kolaborasi yang menemani berjuang dalam menyelesaikan Tugas Akhir ini.

- Teman-teman OSIRIS yang mengisi hari-hari penulis dari awal perkuliahan hingga saat ini

Dalam pengerjaan Tugas Akhir ini, penulis menyadari masih ada kekurangan di dalamnya. Oleh karena itu, penulis sangat terbuka dengan adanya kritik, saran dan pertanyaan yang membangun terkait dengan Tugas Akhir ini. Hal tersebut dapat menjadi masukan untuk penulis agar lebih baik kedepannya, dan juga untuk penelitian selanjutnya. Semoga buku Tugas Akhir ini dapat memberikan manfaat bagi seluruh pembaca.

Surabaya, Juli 2018

Redian Galih Irianti

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB I
PENDAHULUAN

Pada bab ini, akan dijelaskan tentang Latar Belakang Masalah, Perumusan Masalah, Batasan Masalah, Tujuan Tugas Akhir, Manfaat Kegiatan Tugas Akhir dan Relevansi dengan laboratorium RDIB.

1.1. Latar Belakang

Departemen Sistem Informasi merupakan salah satu departemen yang dimiliki Institut Teknologi Sepuluh Nopember yang mempelajari pengembangan dan manajemen suatu sistem informasi, pemodelan bisnis, hingga integrasi sistem informasi. Sebagai suatu departemen yang bergerak dibidang pendidikan, Departemen Sistem Informasi memiliki misi yang digunakan untuk mencapai tujuannya. Salah satu misi yang dimiliki Departemen Sistem Informasi ITS adalah memberikan kualitas pendidikan yang sangat baik dalam hal sistem informasi, mengembangkan dan menerapkan sistem informasi yang dapat membantu masyarakat umum, serta memberikan kepuasan dan kesejahteraan kepada sivitas Sistem Informasi [1].

Banyak usaha yang dilakukan untuk mewujudkan misi kualitas pendidikan dan kepuasan civitas terutama mahasiswa dan dosen. Salah satunya penyusunan jadwal matakuliah. Adanya peraturan akademik yang dimiliki Institut Teknologi Sepuluh Nopember menjadi salah satu pertimbangan dalam penyusunan jadwal mata kuliah semua Departemen yang ada di ITS, termasuk pula Departemen Sistem Informasi. Dalam

peraturan akademik diatur terkait waktu beban Sistem Kredit Semester (SKS). Selain itu juga diatur terkait beban SKS untuk tiap jenjang program studinya [2]. Selain adanya peraturan akademik yang berasal dari ITS, Departemen Sistem Informasi ITS juga memiliki Kurikulum Akademik yang menjadi pedoman pula untuk mengatur proses pembuatan jadwal mata kuliah. Kurikulum akademik tersebut berisi berbagai mata kuliah yang dimiliki oleh Departemen Sistem Informasi disertai dengan jenjang semesternya. Selain itu pada Kurikulum Akademik juga mengatur bobot SKS untuk setiap mata kuliah. Ada pula prasyarat terkait mata kuliah yang harus ditempuh sebelum mengambil mata kuliah selanjutnya [3]. Peraturan Akademik dan Kurikulum Akademik tersebut nantinya juga akan membantu dalam pertimbangan penjadwalan mata kuliah Departemen Sistem Informasi.

Namun meskipun sudah ada peraturan akademik, proses penjadwalan tidak bisa sepenuhnya berjalan secara lancar dan cepat. Hal tersebut karena penjadwalan mata kuliah masih dilakukan secara manual. Aktivitas penjadwalan manual tersebut meliputi pemaparan daftar mata kuliah yang akan dibuka pada semester tersebut, selanjutnya dosen yang juga sekaligus memiliki jabatan di luar Departemen Sistem Informasi seperti Wakil Rektor diberikan hak untuk memilih jadwal mengajar. Setelah request dari para dosen dialokasikan, selanjutnya mempertimbangkan jadwal UPMB agar tidak bentrok. Kemudian jadwal mata kuliah yang lain dimasukkan kedalam timeslot yang belum terisi dengan tetap mempertimbangkan agar kelas tidak bentrok untuk setiap angkatan.

Masih manualnya penjadwalan menimbulkan masalah yang berulang tiap akan membuat jadwal. Masalah tersebut diantaranya adalah waktu yang dialokasikan atau yang dibutuhkan untuk membuat jadwal masih menyita waktu. Untuk penjadwalan mata kuliah bisa memakan waktu sampai 2 minggu, meskipun penjadwalan merupakan kewajiban rutin yang harus dijalani tiap semesternya. Selain itu jadwal mata kuliah yang dihasilkan secara manual juga kurang fleksibel. Hal tersebut akan menyulitkan ketika terjadi perubahan secara mendadak, seperti ketika ada request dari dosen yang berubah saat jadwal sudah tersusun. Masalah lain yang terjadi datang dari pihak mahasiswa, masih terdapat mahasiswa yang mendapatkan jadwal mata kuliah sangat padat dalam sehari. Hal ini tentu membuat mahasiswa menjadi jenuh untuk mengikuti mata kuliah sehingga berpengaruh pula pada performa mahasiswa dalam mengerjakan tugas tiap mata kuliah. Jadwal mata kuliah yang dibuat manual ini masih dianggap belum optimal dikarenakan adanya mahasiswa yang mendapatkan jadwal kuliah full sehari, sedangkan pada hari berikutnya libur. Padahal akan lebih baik jika mahasiswa mendapatkan mata kuliah yang tidak terlalu menumpuk pada satu hari saja.

Sehingga berdasarkan beberapa masalah diatas, perlunya optimasi pada penjadwalan mata kuliah untuk dapat membantu menyelesaikan permasalahan. Beberapa pendekatan terkait metode penyelesaian sudah banyak dikembangkan. Salah satunya adalah motede hyper-heuristic dengan memadukan algoritma variable neighborhood search dan tabu search yang akan digunakan dalam pengerjaan tugas akhir ini. Harapannya

setelah dilakukan optimasi penjadwalan mata kuliah dapat memberikan susunan jadwal mata kuliah yang optimal bagi mahasiswa dan fleksibel terhadap perubahan. Tujuan optimasi penjadwalan mata kuliah ini adalah meminimalkan waktu penjadwalan mata kuliah dan juga meminimalkan jumlah mata kuliah pada semester yang sama dalam satu hari. Tujuan tersebut dicapai dengan tetap memperhatikan beberapa batasan yaitu jumlah mata kuliah, alokasi ruangan, jumlah mahasiswa, dan juga alokasi waktu kuliah.

1.2. Perumusan Masalah

Berdasarkan uraian latar belakang yang telah dijelaskan, maka rumusan masalah dari tugas akhir ini, yaitu :

1. Bagaimana model matematis untuk permasalahan penjadwalan mata kuliah di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya?
2. Bagaimana menerapkan algoritma tabu-variable neighborhood search based hyper-heuristic untuk menyelesaikan permasalahan penjadwalan mata kuliah Departemen Sistem Informasi ITS?
3. Bagaimana perbandingan hasil kinerja antara sistem yang menerapkan algoritma tabu-variable neighborhood search based hyper-heuristic dengan penjadwalan yang dilakukan dengan cara manual?

1.3. Batasan Pengerjaan Tugas Akhir

Batasan pemasalahan dalam tugas akhir ini, yaitu :

1. Tugas akhir ini dilakukan di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember.

2. Data yang digunakan dalam tugas akhir ini yaitu data peserta, data mata kuliahsemester genap dan semester ganjil tahun ajaran 2017-2018 program studi sarjana (S1) Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya.
3. Hasil dari tugas akhir ini ada sistem penjadwalan mata kuliah otomatis berbasis Java

1.4. Tujuan Tugas Akhir

Tujuan yang hendak dicapai dalam pengerjaan tugas akhir ini adalah:

1. Mendapatkan model matematis yang sesuai dengan permasalahn penjadwalan mata kuliah di Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember
2. Mengetahui hasil implementasi algoritma tabu-variable neighborhood search based hyper-heuristic pada permasalahan penjadwalan mata kuliah Departemen Sistem Informasi ITS
3. Mengetahui perbandingan hasil kinerja pendawalan mata kuliah dengan menggunakan algoritma tabu-variable neighborhood search based hyper-heuristic dengan metode manual.

1.5. Manfaat Tugas Akhir

Berikut ini adalah manfaat teoritis bagi keilmuan dan juga manfaat praktis yang bisa didapatkan oleh Departemen Sistem Informasi :

1. Manfaat teoritis bagi keilmuan

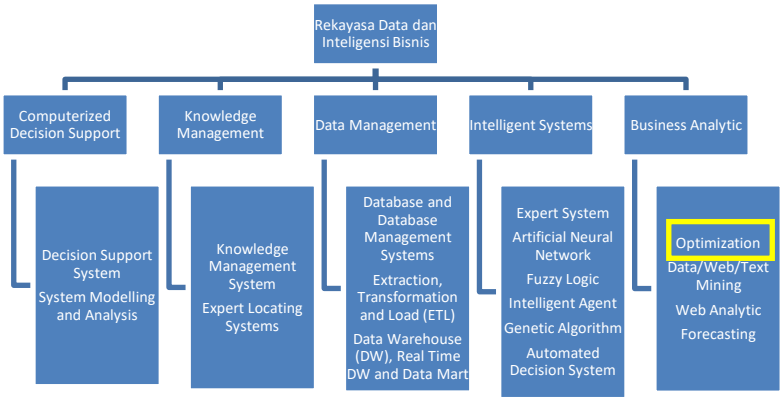
- Dapat dijadikan sebagai pengetahuan dalam bidang optimasi penjadwalan mata kuliah perguruan tinggi.
- Menambah referensi penelitian terkait implementasi metode *tabu-variable neighborhood search based hyper heuristic*.

2. Manfaat praktis bagi Pihak Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember Surabaya

   Diharapkan dapat membantu Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya untuk membuat jadwal kuliah secara otomatis sehingga dapat menghemat waktu dan dapat menghasilkan jadwal perkuliahan yang optimal bagi mahasiswa.

1.6. Relevansi

Penyusunan tugas akhir ini bertujuan untuk memenuhi syarat kelulusan serta sebagai bentuk implementasi disiplin ilmu yang telah didapatkan selama pendidikan perkuliahan di Departemen Sistem Informasi ITS. Dan topik yang diangkat dalam penelitian tugas akhir adalah optimasi yang memiliki relevansi dengan mata kuliah yang dipelajari sebelumnya yaitu Riset Operasi Lanjut. Selain itu topik tersebut juga sesuai dengan bidang ilmu riset operasi yang menjadi cakupan *research roadmap* pada laboratorium Rekayasa Data dan Inteligensi Bisnis (RDIB) yaitu *optimization.*



Gambar 1.1 *Roadmap* Penelitian Laboratorium Rekayasa Data dan Inteligensi Bisnis

(Halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

Dalam Bab ini, akan dijelaskan mengenai penelitian terdahulu dan landasan teori yang digunakan sebagai acuan dalam pengerjaan tugas akhir. Penelitian terdahulu merupakan suatu penelitian yang pernah dilakukan oleh peneliti-peneliti sebelumnya yang diguanakan sebagai acuan tugas akhir. Landasan teori merupakan teori-teori yang berhubungan dengan pengerjaan tugas akhir.

2.1. Studi Sebelumnya

Pada sub bab ini akan diterangkan mengenai beberapa penelitian terdahulu yang telah dilakukan dan memiliki relevansi dengan tugas akhir ini. Penelitian terdahulu tersebut dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No	Penelitian Terdahulu	
1.	Judul Penelitian	Heuristic Approaches For University Timetabling Problems [1]
	Tahun Penelitian	2006
	Nama Peneliti	Salwani Abdullah
	Penjelasan Singkat	Pada penelitian ini menggunakan algoritma *Variable Neighborhood Search* untuk mencari solusi. Didalam pengerjaannya, penulis juga

No	Penelitian Terdahulu	
		mengkombinasikan berbagai pendekatan agar dapat mencapai solusi optimal. Pada tahap awal dilakukan penentuan solusi awal yang akan digunakan untuk acuan iterasi berikutnya. Setelah itu menentukan *Neighborhood Structure*, kemudian menentukan *acceptance criteria* untuk mempermudah menemukan solusi terbaik. Dalam praktiknya penulis juga menggunakan pendekatan *Tabu Search* yang digunakan untuk menyimpan *neighborhood structure* yang tidak menghasilkan solusi optimal kedalam *tabu list*. Hasil dari penelitian ini menunjukkan bahwa dengan menggabungkan *variable neighborhood search* dan *tabu search* dapat menghasilkan solusi yang *feasible* dan juga optimal. Dataset yang digunakan merupakan dataset Socha yang berisi 3 kategori yaitu *small, medium,* dan *large.*
	Keterkaitan dengan Tugas Akhir	Algoritma yang digunakan untuk pengerjaan thesis diatas yang akan diadopsi juga pada penelitian kali ini, yaitu *variable neighborhood search* dan *tabu search.* Selain itu dataset Socha juga akan digunakan sebagai masukan data

No	Penelitian Terdahulu	
		pada penelitian kali ini.
2.	Judul Penelitian	The Effect Of Neighborhood Structures On Tabu Search Algorithm In Solving Course Timetabling Problem [2]
	Tahun Penelitian	2009
	Nama Peneliti	Cagdas Hakan Aladag, Gulsum Hocaoglu, Murat Alper Basaran
	Penjelasan Singkat	Penelitian [2] mendapatkan solusi dengan melakukan dua kombinasi penyusunan *neighborhood structure*. Penelitian ini mengambil studi kasus Departemen Statistik Universitas Hacettepe. Pendekatan yang dilakukan adalah menggunakan *neighborhood structure* dan juga *Tabu List.* Untuk mendapatkan *neighborhood structure* dilakukan dengan menggunakan dua metode perpindahan yaitu *simple move* dan *swap move*. Kemudian masing-masing metode perpindahan tersebut akan dikombinasikan dengan pendekatan *Tabu Search*. Hasil penelitian menunjukkan solusi lebih optimal didapatkan dengan menggabungkan *Tabu Search* dengan *neighborhood structure* yang menggunakan metode perpindahan

No	Penelitian Terdahulu	
		simple move
	Keterkaitan dengan Tugas Akhir	Pendekatan dan algoritma yang digunakan dapat menjadi referensi untuk diterapkan dalam pengerjaan penelitian ini. Penentuan *neighborhood structure* dengan mengikuti langkah-langkah *simple move* dan *swap move*.
3.	Judul Penelitian	A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem [3]
	Tahun Penelitian	2002
	Nama Peneliti	Olivia Rossi-Doria, Michael Sampels, Mauro Birattari, Marco Chiarandini, Marco Dorigo, Luca M. Gambardella, Joshua Knowles, Max Manfrin, Monaldo Mastrolilli, Ben Paechter, Luis Paquete, and Thomas St¨utzle
	Penjelasan Singkat	Penelitian ini terkait penyelesaian masalah penentuan jadwal perkuliahan. Masalah penentuan jadwal yang dipertimbangkan disini adalah pengurangan masalah jadwal mata kuliah perguruan tinggi yang khusus. Hal tersebut mengacu pada penelitian Ben Paechter untuk merefleksikan aspek

No	Penelitian Terdahulu	
		masalah *real time* aktual dari *Napier University*. Metode yang dibandingkan merupakan algoritma *meta heuristics* yaitu *Evolutionary Algorithms, Ant Colony Optimization, Iterated Local Search, Simulated Annealing,* dan *Tabu Search*. Hasil penelitian menunjukan performa yang mirip untuk setiap algoritma yang dilakukan. Namun pada kasus contoh yang *large*, *Tabu Search* dapat menghasilkan solusi feasible sebesar 8% dari seluruh percobaan. Pada penelitian ini juga dijelaskan bahwa penggunaan satu algoritma saja dirasa kurang untuk dapat menghasilkan performa yang baik, karena setidaknya harus melakukan dua tahapan untuk mendapatkan hasil optimal.
	Keterkaitan dengan Tugas Akhir	Pada penelitian ini membandingkan beberapa algoritma heuristik salah satunya adalah *Tabu Search* untuk melihat performa terbaik, hal tersebut menjadi pertimbangan dan acuan dalam Tugas Akhir ini dalam hal penentuan data set dan juga penggunaan algoritma *Tabu Search.*
4.	Judul Penelitian	Implementasi Algoritma Genetik-Tabu Search dalam Optimasi Penjadwalan

No	Penelitian Terdahulu	
		Perkuliahan [4]
	Tahun Penelitian	2016
	Nama Peneliti	Rusianah, M. Aziz Muslim, Sholeh Hadi Pramono
	Penjelasan Singkat	Pada penelitian [4] pencarian solusi dilakukan dengan menggabungkan dua algoritma yaitu Genetika dan *Tabu Search.* Penggunaan algoritma *Tabu Search* dilakukan memperbaiki performansi *local search* dengan memanfaatkan penggunaan struktur memori. Dengan menggabungkan dua algoritma tersebut waktu yang diperlukan untuk mencapai nilai *fitness* 1 lebih cepat dibandingan dengan hanya menggunakan algoritma genetika saja. Begitu pula dengan iterasi generasi yang didapat dengan juga lebih sedikit jika menggunakan kombinasi dua algoritma.
	Keterkaitan dengan Tugas Akhir	Pencarian solusi dilakukan dengan mengkombinasi dua pendekatan. Hal tersebut menjadi referensi dalam penelitian ini karena menggabungkan dua pendekatan pula, terutama penggunaan *Tabu Search.*

No	Penelitian Terdahulu	
5.	Judul Penelitian	Optimasi Penjadwalan Ujian Otomatis dengan Menggunakan Algoritma Greedy-Late Acceptance-Hyper Heuristic [5] Optimasi Penjadwalan Menggunakan Algoritma Greedy Simulated Annealing Hyper Heuristic [6] Optimasi Penjadwalan Ujian Otomatis Dengan Menggunakan Algoritma Greedy Hill Climbing Hyper-Heuristic [7]
	Tahun Penelitian	2018
	Nama Peneliti	Putri Cahyaning Bwananesia, Dian Kusumawardani, Gigih Yudha Hutama
	Penjelasan Singkat	Pada beberapa penelitian terkait penjadwalan ujian dilakukan dengan mengambil studi kasus Departemen Sistem Informasi ITS. Penyelesaian masalah dilakukan dengan berbagai algoritma berbeda, namun tetap berbasis *hyper-heuristic*. Pada penjadwalan ujian terdapat batasan yang juga harus diperhatikan baik itu *hard* maupun *soft*. Dari ketiga penelitian, performa terbaik dilakukan dengan menggunakan algoritma Greedy Simulated Annealing

No	Penelitian Terdahulu	
		Hyper Heuristic karena dengan iterasi yang sedikit dapat memperoleh hasil yang cukup baik.
	Keterkaitan dengan Tugas Akhir	Penelitian diatas menyelesaikan permasalahan dengan algoritma berbasis *hyper-heuristic.* Hal tersebut dapat menjadi referensi untuk penelitian ini terkait implementasi *hyper-heuristic* pada penyelesaian permasalahan penjadwalan mata kuliah.
6.	Judul Penelitian	Penjadwalan Mata Kuliah Menggunakan Algoritma Genetika di Jurusan Sistem Informasi ITS [8]
	Tahun Penelitian	2013
	Nama Peneliti	Wiga Ayu Puspaningrum, Arif Djunaidy, dan Retno Aulia Vinarti
	Penjelasan Singkat	Penelitian ini mengambil studi kasus Jurusan Sistem Informasi ITS. Penyelesaian masalah penjadwalan mata kuliah ini meggunakan algoritma Genetika dengan mempertimbangkan beberapa batasan yang ada di jurusan seperti ketersediaan dosen, mahasiswa yang mengambil mata kuliah, ketersediaan waktu kuliah dan juga

No	Penelitian Terdahulu	
		ketersediaan ruang kelas.
	Keterkaitan dengan Tugas Akhir	Penelitian ini menggunakan studi kasus yang sama dengan Tugas Akhir yaitu Departemen Sistem Informasi ITS, hal tersebut dapat menjadi referensi dalam menentukan batasan dan juga identifikasi data set yang dibutuhkan.

2.2. Dasar Teori

Pada sub bab ini akan dijabarkan mengenai dasar teori yang digunakan untuk mendukung pengerjaan tugas akhir ini.

2.2.1. Penjadwalan

Penjadwalan menjadi salah satu aktivitas yang banyak dilakukan pada institusi pendidikan ataupun institusi yang lainnya. Hingga saat ini, permasalahan masih sering terjadi dalam proses penjadwalan. Salah satu masalahnya adalah terlalu banyak memakan waktu dalam pembuatan jadwal yang dilakukan secara manual. Penjadwalan sendiri dapat didefinisikan sebagai sebuah permasalahan yang memiliki empat parameter: himpunan berhingga waktu (T), himpunan berhingga sumber daya (R), himpunan berhingga pertemuan (M), himpunan berhingga batasan (C). Dalam hal ini, masalah yang dimaksud adalah terkait pengalokasian waktu dan sumber daya terhadap suatu pertemuan untuk memenuhi batasan-batasan sebanyak mungkin [9].

Penjadwalan merupakan sebuah permasalahan menarik yang telah diteliti selama lebih dari empat dekade di berbagai bidang ilmu, khususnya di bidang riset operasi dan kecerdasan buatan. Penjadwalan adalah permasalahan optimasi kombinatorik yang didefinisikan oleh By Lawler sebagai studi matematis untuk mencari solusi optimal pada penyusunan, pengelompokan, pengurutan atau pemilihan objek diskrit yang biasanya berjumlah terbatas *(finite number)*[10].

Terdapat pula pendapat dari Schaerf yang menyatakan bahwa permasalahan terkait penjadwalan dapat dikategorikan menjadi beberapa bagian, diantaranya adalah permasalahan penjadwalan sekolah, permasalahan penjadwalan per mata kuliah, dan permasalahan penjadwalan ujian. Berdasarkan kategori tersebut, Schaerf berpendapat tidak ada perbedaan yang cukup mencolok dari ketiganya. Sebuah permasalah penjadwalan dapat terdiri dari salah satu atau diantara dua kategori diatas [11].

Permasalahan penjadwalan mata kuliah menjadi masalah yang muncul secara berkala pada sebuah perguruan tinggi. Masalah umum terdiri dari penetapan agenda seperti kelas, ceramah, tutorial, dan lain-lain menjadi sejumlah *timeslots* terbatas sehingga dapat memenuhi setiap batasan yang ada. Batasan yang harus dipenuhi biasanya digolongkan menjadi dua yaitu *hard constraint* dan *soft constraint.* Model yang dibuat tidak boleh melanggar *hard constraint* yang ada. Sedangkan hasil implementasi akan lebih baik ketika dapat memenuhi *soft constraint*, namun tetap dapat diterima dengan *penalty* tertentu [3].

Adapula Carter dan Laporte yang mendefinisikan penjadwalan mata kuliah sebagai masalah penugasan multi dimensi dimana siswa, dosen dipetakan pada mata kuliah dan kelas tertentu, serta penentuan waktu yang tidak boleh bertabrakan [12].

Permasalahan penjadwalan mata kuliah diklasifikasikan sebagai NP-*hard*[9]. Hal tersebut berarti permasalahan penjadwalan mata kuliah tidak dapat diselesaikan dengan mudah menggunakan metode konvensional waktu penyelesaian yang dibutuhkan tergantung pada seberapa besar permasalahan yang diselesaikan.

2.2.2. Formulasi Model Matematis Penjadwalan Mata Kuliah

Dalam melakukan penyelesaian permasalahan penjadwalan mata kuliah, terdapat beberapa hal yang harus diperhatikan sebelum melakukan implementasi algoritma. Hal tersebut biasa disebut dengan istilah permodelan matematis. Yang pertama terkait variabel keputusan yang berfungsi untuk menggambarkan tingkatan aktifitas perusahaan, biasanya berupa simbol matematika. Berikut variabel keputusan yang biasa digunakan dalam permasalahan penjadwalan mata kuliah [1] :

- N : jumlah mata kuliah yang diberi label {e_1, e_2, e_3, …., e_N)
- T : jumlah *timeslots* yang tersedia
- R : jumlah ruangan yang tersedia
- F : satu set fasilitas dalam ruangan
- M : jumlah mahasiswa
- D : jumlah dosen yang tersedia

- P : nilai pinalti yang dihasilkan dari *soft constraint*

Selanjutnya yang harus diperhatikan adalah fungsi tujuan dari permasalahan penjadwalan mata kuliah. Fungsi tujuan menjelaskan tujuan yang ingin dicapai dari penyelesaian permasalahan. Fungsi tujuan selalu memiliki salah satu target yaitu memaksimumkan atau meminimumkan suatu nilai. Fungsi tujuan pada permasalahan penjadwalan mata kuliah ini adalah meminimumkan nilai pinalti dari *soft constaint*s pada solusi yang *feasible.*Nilai pinalti diberikan untuk setiap *soft constraint* yang ada. Setelah itu, pengecekan dilakukan pada setiap mahasiswa terkait pelanggaran *soft consraint.* Setiap mahasiswa yang melanggar *soft constraint* terkait, maka akan diberikan pinalti. Nilai 1 diberikan untuk setiap *soft constraint* dengan berdasarkan time slot yang dilanggar. Akumulasi dari nilai pinalti tersebut yang akan menentukan nilai dari solusi yang didapatkan. Dapat dirumuskan seperti dibawah ini [1]:

$$P = P_1 + P_2 + P_3$$

Yang terakhir, solusi yang akan dibuat harus mempertimbangkan sumber daya terbatas yang ada pada *hard constraint* dan *soft constraint*. Dalam banyak kasus, penentuan solusi yang layak dilakukan untuk memenuhi *hard constraint*. Sedangkan *soft constraint* merupakan pilihan, solusi yang dihasilkan sebisa mungkin memenuhi sebanyak mungkin *soft constraint* yang ada [13].

Burke dan Petrovic membagi batasan untuk pertimbangan pembuatan solusi menjadi dua, seperti yang sudah banyak dibahas sebelumnya. Batasan tersebut adalah *hard constraint*, merupakan batas-batas yang harus diterapkan pada

penjadwalan mata kuliah dan harus dipenuhi [14]. Beberapa contoh *hard constraint* yang umum digunakan pada permasalahan penjadwalan mata kuliah diantaranya [12] :

- Mahasiswa tidak bisa mengikuti perkuliahan pada mata kuliah berbeda dalam waktu yang sama
- Dosen tidak dapat mengajar mata kuliah berbeda dalam waktu yang sama
- Setiap kelas hanya dapat digunakan untuk satu jadwal mata kuliah pada waktu yang sudah disediakan
- Kapasitas ruangan harus kurang dari atau sama denggan jumlah siswa yang akan dialokasikan pada kelas tersebut

Sedangkan *soft contraint* didefinisikan sebagai batas-batas terkait alokasi sumber daya yang jika dilanggar masih dapat menghasilkan solusi yang layak, tetapi sedapat mungkin untuk dipenuhi. Semakin banyak *soft constraint* yang berhasil dipenuhi maka akan memuaskan pihak terkait [14]. Sebagai contoh *soft constraint* adalah mahasiswa hanya terjadwalkan maksimal dua mata kuliah dalam satu hari.

Terkait dengan *hard constraint* dan *soft constraint*, tujuan umum yang ingin dicapai dalam permasalahan penjadwalan adalah memenuhi semua *hard constraints* dan meminimalisasi pelanggaran terhadap *soft constraints*, yang dapat bervariasi tergantung dari kebijakan masing-masing instansi. Batasan-batasan ini telah diteliti oleh Burke [15]. Solusi yang dapat memenuhi semua *hard constraints* disebut sebagai solusi yang mungkin *(feasible solution)*. Apabila solusi tersebut juga dapat memenuhi semua *soft constraints* (tidak ada pelanggaran

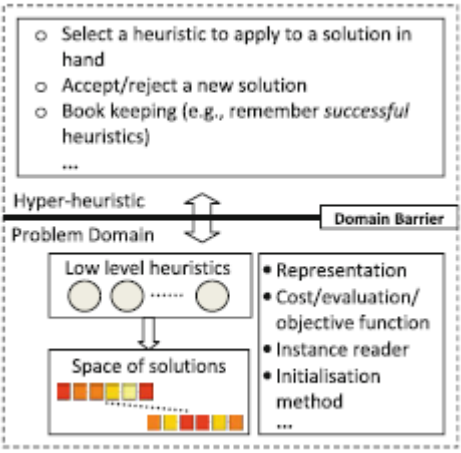
apapun terhadap *soft constraints*), maka solusi itu disebut sebagai solusi yang sempurna *(perfect solution)*.

2.2.3. Algoritma *Hyper-Heuristic*

Hyper-heuristic merupakan metodologi pencarian tingkat tinggi yang melakukan pencarian diatas ruang *heuristic*. Ide dasar dari algoritma *hyper-heuristic* adalah gagasan untuk menggabungkan heuristik yang berbeda dengan tujuan memanfaatkan kekuatan masing-masing heuristik sejak tahun 1960an. Sejak saat itu, minat terhadap *hyper-heuristic* menjadi meningkat. Sebuah studi teoritis menunjukkan bahwa penggabungan heuristik dapat menyebabkan pencarian lebih cepat secara eksponensial jika dibanding dengan menggunakan heuristik mandiri pada beberapa kasus [13].

Sebuah pilihan heuristik umumnya menggabungkan pilihan heuristik kemudian memindahkan metode penerimaan dibawah kerangka iteratif. Pada setiap langkah, *low level heuristic* digunakan untuk memodifikasi solusi yang ada[13].

Gambar 2.1 menunjukkan *framework* dari *hyper-heuristic.* Pada *framework* ini menunjukkan bahwa algoritma *hyper-heuristic* menggunakan *problem domain barrier* yang membuat strategi *hyper-heuristic* tidak bersinggungan langsung dengan *space of solution*. Sebaliknya, algoritma *hyper-heuristic* hanya bersinggungan dengan *heuristic* pada level yang lebih rendah *(low level heuristics)* [13]. Dengan *search space* yang lebih tinggi ini, metode *hyper-heuristic* tidak tergantung pada *problem domain* tertentu saja, sehingga tidak diperlukan parameter tuning untuk setiap problem domain secara manual. Mekanisme hyper-heuristic mampu melakukan otomasi proses parameter tuning ini [16].



Gambar 2.1 ***Framework Hyper-heuristic***

2.2.4. Algoritma *Tabu Search*

Kata tabu atau *taboo* berasal dari Tongan, sebuah bahasa Polynesia, yang mana telah digunakan oleh orang aborigin pulau Tongan yang memiliki arti barang yang tidak bisa disentuh karena disakralkan. Pertama kali ide tentang *Tabu Search* muncul oleh Fred Glover tahun 1986 yang setuju dengan metode *local search* untuk memecahkan masalah optimasi [17]. Glover dan Laguna mendefinisikan *Tabu Search* sebagai salah satu metode *meta-heuristic* yang memandu prosedur pencarian heuristik lokal untuk mengeksplorasi ruang solusi di luar optimalitas lokal. Masalah yang ditakutkan akan muncul adalah pencarian solusi yang terjebak dalam area *local optima*. Namun untuk mencegah hal tersebut, dilakukan dengan cara mempertahankan daftar tabu berisi langkah-langkah yang memenuhi beberapa kriteria pembatasan tabu. Langkah-langkah yang sebelumnya sudah masuk kedalam daftar tabu tersebut dilarang untuk diikutkan

dalam iterasi tertentu, biasa disebut *tabu tenure*. *Tabu tenur* menentukan berapa lama sebuah langkah tetap tabu. Namun mekanisme yang biasa disebut *aspiration criterion* kadang-kadang digunakan untuk mengganti status tabu dari sebuah pergerakan. *Aspiration criterion* yang umum adalah nilai *fitness* yang lebih baik(peningkatan fungsi biaya) dimana pergerakan tabu diubah menjadi langkah non-tabu saat menghasilkan solusi yang lebih baik [1].

Tabu Search merupakan metode optimasi yang menerapkan sistem pemanfaatan *memory*. *Tabu search* menggunakan struktur memori yang disebut *Tabu List*, digunakan untuk menyimpan langkah solusi yang pernah ditemui selama iterasi berjalan agar proses pencarian tidak berulang-ulang pada area solusi yang sama. Selain itu juga untuk menuntun proses pencarian ke solusi-solusi yang belum pernah dikunjungi sebelumnya [4]. Selama proses optimasi, pada setiap iterasi, solusi yang akan dievalusi akan dicocokkan terlebih dahulu dengan isi *tabu list*. Apabila solusi tersebut sudah ada pada *tabu list*, maka solusi tersebut tidak akan dievaluasi lagi pada iterasi berikutnya. Apabila sudah tidak ada lagi solusi yang tidak menjadi anggota *tabu list*, maka nilai terbaik yang baru saja diperoleh merupakan solusi yang sebenarnya [17].

Algoritma *Tabu Search* sederhana memiliki tiga strategi utama diantaranya (1) *forbidding strategy, (2) freeing strategy,* dan (3) *short-term strategy*. (1) *Forbidding strategy* yaitu mengontrol apa saja parameter yang akan masuk ke dalam tabu list. (2) *Freeing strategy* yaitu mengontrol apa saja dan kapan yang keluar dari *tabu list*. (3) *Short-termstrategy* yaitu

mengatur kolaborasi antara *forbidding strategy* dan *freeing strategy*[18].

Konsep algoritma *Tabu Search* secara umum dapat dituliskan seperti Kode 2.1 berikut [4]:

```
1) Bangkitkan solusi awal yang layak, misalkan s, secara
   acak atau menggunakan metode heuristik tertentu.
2) BiayaOptimum = Biaya (s)
3) s'' = s {s'' adalah solusi terbaik yang diperoleh}
4) TabuList = null
5) Repeat
a. V* = himpunan solusi yang merupakan tetangga dari s
   yang memenuhi criteria aspirasi atau tidak berada dalam
   TabuList
b. Pilih s* {s* adalah solusi yang memiliki biaya minimum
   di dalam V*}
c. Simpan move yang berlawanan kedalam TabuList, yang
   mengubah s ke s*
d. s = s*
   if (Biaya (s) < BiayaOptimum) then
   s* = s
   BiayaOptimum – Biaya (s)
   End
   Until (KriteriaBerhenti = true)
```

Kode 2.1 *Pseudocode* Algoritma *Tabu Search*

Algoritma ini akan mencegah struktur lingkungan terbaru yang belum berjalan dengan baik untuk dipilih pada iterasi berikutnya, sehingga pencarian dapat diarahkan ke area pencarian lainnya [1].

2.2.5. Algoritma *Variable Neighborhood Search*

Variable Neighborhood Search (VNS) diperkenalkan oleh Mladevonic dan Hansen. Hal ini didasarkan pada strategi menggunakan lebih dari satu struktur lingkungan dan mengubah struktur tersebut secara sistematis selama pencarian lokal. Hal ini membantu VNS untuk menjelajahi lingkungan yang jauh dari solusi saat ini dan beralih ke solusi baru [19].

Variable Neighborhood Search terdiri dari tiga tahapan : (1) *Shaking* yaitu mengacak solusi yang ada, (2) *Local Search* yaitu mencari solusi baru pada area solusi, dan (3) *Move* yaitu tindakan untuk solusi baru yang dihasilkan, jika lebih baik dari sebelunya maka akan menggantikan solusi yang lama[19][20]. Pencarian lokal diterapkan berulang kali untuk mendapatkan *local optimum* dari solusi saat ini. Pada awalnya pendekatan VNS dasar merupakan metode turunan yang tidak menerima solusi yang memburuk untuk keluar dari pilihan lokal karena struktur lingkungan bervariasi secara teratur. Struktur lingkungan dapat dilakukan selama pencarian, hal tersebut dikarenakan *local optimum* dalam satu struktur lingkungan belum tentu menjadi pilihan lokal pada struktur lingkungan lain [19].

Untuk pengaplikasian algoritma *Variable Neighborhood Search* secara umum terdapat beberapa hal yang harus diperhatikan, diantaranya [1] :

1) *Initial Solution* : Penentuan solusi awal dapat dilakukan dengan beberapa pendekatan heuristik. Tujuan ditetapkannya solusi awal adalah untuk acuan dalam proses iterasi pada tahapan selanjutnya. Pada

penelitian [1] menggunakan *constructive heuristic* untuk membuat solusi awal.

2) *Neighborhood Structure* : penentuan struktur lingkungan digunakan untuk membantu dalam proses *local search.* Struktur yang dihasilkan akan digunakan untuk mendapatkan solusi baru.
3) *Acceptance Criteria* : penentuan *acceptance criteria* ditujukan untuk mendapatkan solusi yang lebih optima. *Acceptance criteria* ini nantinya yang akan menjadi acuan apakah sebuah solusi baru dapat diterima atau tidak. Pada penelitian [1] menggunakan dua *acceptance criteria* yaitu *descent* dan *exponential monte carlo.*
4) *Local Search* : merupakan pencarian solusi baru pada area solusi. Pencarian solusi baru ini memperhatikan aspek-aspek yang telah ditentukan sebelumnya seperti *neighborhood structure, acceptance criteria,* dan *stopping criterion.*
5) *Stopping Criterion* : digunakan sebagai acuan kapan sistem akan berhenti melakukan iterasi pencarian solusi. Biasanya didefinisikan sebagai *eval* > *Max_eval* , dimana *eval* merupakan evaluasi yang terjadi. Nilai *Max_eval* diberikan selalu konstan. Pada penelitian [1] nilai *Max_eval* diatur sebesar 200.000. *Stopping Criterion* ditentukan saat awal akan menjalankan sistem.

Secara umum, VNS didasarkan pada tiga aturan: (1) Solusi optimum global yaitu optimum lokal di semua struktur lingkungan yang mungkin, (2) Solusi optimum lokal yaitu sehubungan dengan struktur lingkungan tunggal yang belum

tentu menjadi solusi pada struktur lingkungan lainnya, (3) Untuk banyak malasah, pilihan lokal dari satu atau lebih struktur lingkungan relatif dekat satu sama lain [21].

2.2.7. Algoritma *Tabu-Variable Neighborhood Search*

Dengan menggabungkan dua algoritma diharapkan dapat menghasilkan solusi yang lebih optimal. Algoritma *Tabu Search* digunakan untuk menangani struktur lingkungan yang tidak mengarah pada solusi baru yang diterima. Pembatasan tabu diterapkan dimana struktur lingkungan n_k akan menjadi tabu jika nilai solusi baru *s* lebih besar dari nilai solusi lama dan solusinya ditolak oleh *accaptance criteria* yang berlaku. Algoritma *Tabu-Variable Neighborhood Search* dapat digambarkan berupa pseudocode seperti Kode 2.2 dibawah ini [1] :

```
Initialisation:
   (1) Select the set of neighbourhood structures nk, k=1,…,K that
       will be used in the random descent local search; generate
       an initial solution s; choose a termination criterion;
   (2) Record the best solution sbest ← s and f(sbest) ← f(s);

Repeat until the termination criterion is met:
   (1) Set k ← 1;
   (2) Until k = K, repeat:
   (a) Shaking: Generate a random solution s' from the nk
       neighbourhood of s (s'∈ nk(s));
   (b) Local search: Apply a random-descent local search to s'
       until local optimum s" is obtained;
   (c) Move or not:
       if ((f(s") is better than incumbent solution s) or (f(s")
       is accepted by the acceptance criterion)) then
            s ← s";
            set k  ← 1;
            while k is in the tabulist and k < K
                 k ← k+1;
            continue the search with nk;
       else
            insert k to the tabulist;
            set k ← k+1;
            increase the tabu length by 1;
            if tabu length > tabu tenure
                 release the first neighbourhood structure from the
                 tabu list;
            while k is in the tabulist and k < K
                 k ← k+1;
```

Kode 2.2 ***Pesudocode Tabu-Variable Neighborhood Search***

BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai alur metodologi yang akan dilakukan dalam tugas akhir ini. Metodologi ini juga digunakan sebagai pedoman untuk melaksanakan tugas akhir agar terarah dan sistematis.

3.1. Metodologi Penelitian

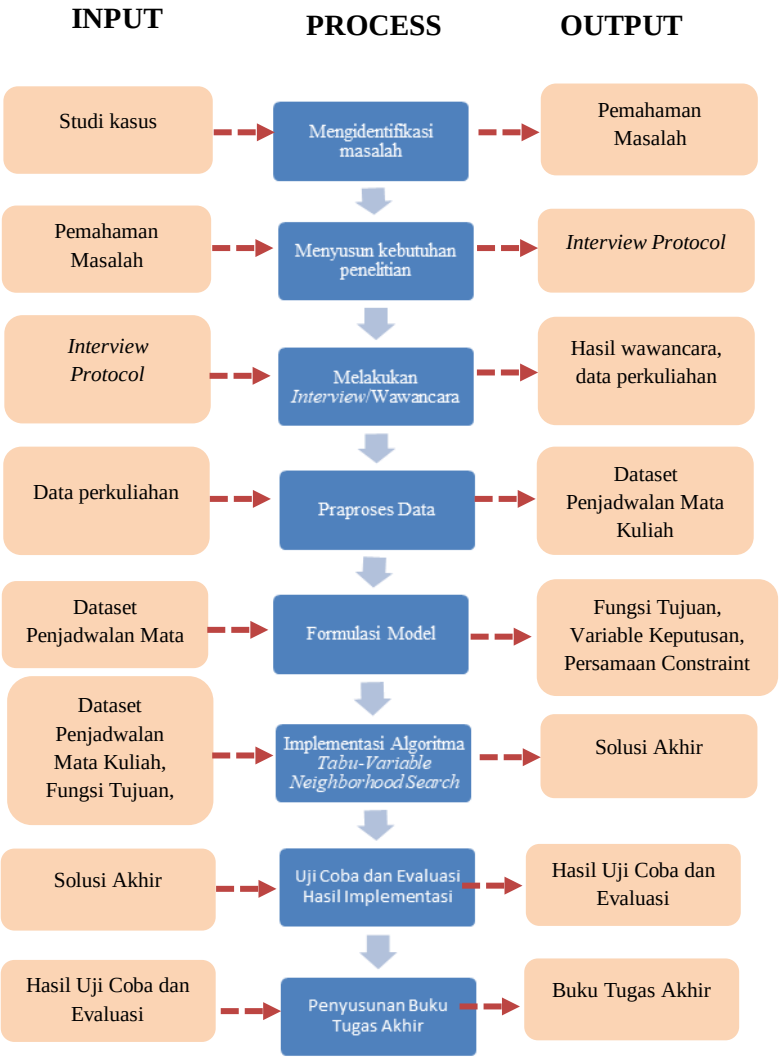
Diagram Metodologi dari Tugas Akhir ini dapat dilihat pada Gambar 3.1 akan menjelaskan alur pengerjaan Tugas Akhir disertai *input* dan *output* yang digunakan.

3.2. Tahapan Pelaksanaan Tugas Akhir

Tahapan pelaksanaan tugas akhir akan menjelaskan terkait segala sesuatu yang akan dikerjakan oleh penulis atau merupakan langkah-langkah pengerjaan tugas akhir.

3.2.1. Identifikasi Masalah

Pada tahap ini adalah tahapan untuk memahami studi kasus yang ada yaitu terkait penjadwalan matakuliah pada perguruan tinggi. Pada kasus kali ini, objek yang digunakan adalah Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember. Diketahui hingga saat ini proses pembuatan jadwal yang dilakukan masih manual, sehingga penyusunan jadwal masih memerlukan waktu yang tidak singkat. Pada tahap ini pula ditetapkan tujuan dari pemecahan masalah dan batasan pengerjaan masalah. Untuk mendapatkan informasi lebih banyak terkait permasalahan penjadwalan mata kuliah, maka pada tahap ini juga dilakukan studi literatur dari paper, jurnal, maupun sumber bacaan yang lain. Dari tahap ini mendapatkan hasil berupa algoritma yang cocok digunakan, yaitu dengan menggabungkan dua algoritma untuk mendapatkan hasil yang lebih optimal.



Gambar 3.1 Metodologi Penelitian

3.2.2. Menyusun Kebutuhan Penelitian

Setelah menentukan tujuan, batasan masalah, dan juga algoritma yang digunakan, maka dilanjutkan ke tahap menyusun kebutuhan penelitian. Pada tahap ini segala informasi yang dibutuhkan untuk menunjang penelitian akan didata sehingga memudahkan dalam langkah selanjutnya. Kebutuhan tersebut dapat berupa data maupun informasi yang diperlukan. Hasil dari tahapan ini adalah *interview protocol* terkait kebutuhan informasi penjadwalan mata kuliah di Departemen Sistem Informasi. Selain itu ada pula daftar kebutuhan data untuk penelitian.

3.2.3. Melakukan *Interview*/Wawancara

Tahap selanjutnya adalah melakukan wawancara pada narasumber terkait dengan kebutuhan informasi yang telah dibuat dalam *interview protocol*. Pada tahap ini penulis melakukan wawancara kepada Ibu Feby Artwodini Muqtadiroh, S.Kom, M.T selaku Sekretaris Program Studi Sarjana (S1) Sistem Informasi, Departemen Sistem Informasi. Narasumber dipilih karena merupakan pihak yang terlibat langsung dalam pembuatan jadwal mata kuliah pada Departemen Sistem Informasi. Hasil dari tahapan ini adalah informasi terkait kondisi terkini prosess penjadwalan pada Departemen Sistem Informasi dan juga jadwal kuliah Program Studi Sarjana (S1) Sistem Informasi pada Semester Genap dan Semester Ganjil, Tahun Ajaran 2017/2018.

3.2.4. Praproses Data

Data yang digunakan pada tugas akhir ini adalah data terkait dengan kebutuhan penjadwalan mata kuliah yang dilaksanakan pada semester ganjil dan genap tahun ajaran 2017 – 2018 pada

Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember Surabaya. Data yang didapatkan antara lain : daftar mata kuliah, daftar dosen, daftar ruangan, dan alokasi waktu. Alokasi waktu yang digunakan adalah 3 sesi per hari selama 4 hari senin hingga kamis, dan 2 sesi pada hari jumat. Jadwal mata kuliah yang telah didapat kemudian dilakukan pra-proses. Tahap ini dilakukan untuk membuat data mentah menjadi siap diolah menggunakan algoritma *tabu-variable neighborhood search*. Data yang masih mentah disesuaikan dengan format dataset Socha [22] yang terdiri dari matrik mahasiswa/mata kuliah, ruangan/fasilitas, dan mata kuliah/fasilitas yang disusun secara vertikal.

3.2.5. Formulasi Model

Pada tahap ini model awal dibuat untuk menyelesaikan permasalahan berdasarkan beberapa variabel seperti :

1) Variabel Keputusan
2) Fungsi Tujuan
   Meminimalkan nilai penalti terhadap *soft constraint*
3) Fungsi Batasan
   Fungsi batasan terdiri dari *hard constraint* dan *soft constraint* dimana adanya *hard constraint* harus dipenuhi keseluruhan, dan sebisa mungkin memenuhi *soft constraint* yang ada.

Model yang dibuat akan mengacu pada pembahasan tinjauan pustaka poin 2.2.2 yang disesuaikan dengan permasalahan penjadwalan mata kuliah Departemen Sistem Informasi.

3.2.6. Implementasi Algoritma *Tabu-Variable Neighborhood Search*

Pada tahap implementasi algoritma *Tabu-Variable Neighborhood Search*, data yang telah mengalami pra-proses dijadikan sebagai masukan. Tahapan dimulai dengan menentukan solusi awal yang nantinya akan dipakai sebagai acuan untuk iterasi berikutnya. Setelah menentukan solusi awal, dilanjutkan dengan menyusun struktur lingkungan. Struktur lingkungan digunakan sebagai pola acak yang akan menghasilkan solusi saat iterasi. Selain menentukan solusi awal, hal yang perlu diperhatikan juga adalah menentukan *acceptance criteria* untuk syarat bagaimana solusi dapat diterima menjadi solusi baru menggantikan solusi yang lama. Hal terakhir yang harus diinisiasi adalah *stopping criteria*, yaitu kriteria yang akan membuat sistem berhenti mencari solusi baru. Setelah hal-hal diatas dipersiapkan, maka dilanjutkan dengan proses iterasi yang akan menghasilkan solusi baru. Peran *Tabu Search* akan diimplementasikan. Proses iterasi pada *local search* dilakukan dengan memperhatikan struktur lingkungan dan juga *acceptance criteria*. Ketika solusi baru didapatkan dan tidak melanggar *acceptance criteria*, maka iterasi akan dilanjutkan. Namun jika solusi baru yang didapatkan lebih buruk atau sama dengan solusi lama, maka struktur lingkungan yang terlibat akan dimasukkan kedalam *Tabu List*, sehingga tidak akan digunakan lagi untuk penentuan solusi baru pada iterasi selanjutnya.Jika terdapat stuktur lingkungan yang menghasilkan solusi tidak optimal, maka akan dimasukkan lagi kedalam tabulist dan menggantikan anggota tabulist yang

sebelumnya. Solusi yang sebenarnya didapat saat sudah mencapai *stopping criteria*.

3.2.7. Uji Coba dan Evaluasi Solusi Implementasi

Pada tahapan ini akan dilakukan pengujian serta evaluasi algoritma *Tabu-Variable Neighborhood Search* yang telah dirancang sebelumnya. Proses uji coba dilakukan untuk melakukan verifikasi dan validasi algoritma *Tabu-Variable Neighborhood Search* yang dirancang. Apakah algoritma yang dirancang sesuai dengan cara kerja, dan apakah sudah tepat sesuai kebutuhan. Jika terdapat kesalahan atau kekurangan, maka akan dilakukan evaluasi dan perbaikan. Selain itu, pada tahap ini juga dilakukan perbandingan hasil antara optimalisasi dan hasil manual yang digunakan sebelumnya. Perbandingan dilakukan dengan melihat akumulasi nilai pinalti yang dihasilkan dari solusi sistem dan juga nilai pinalti yang dihasilkan dari jadwal manual. Untuk jadwal manual, perhitungan pinalti akan dilakukan menggunakan perhitungan fungsi tujuan dengan membuat input data berdasarkan hasil jadwal manual yang disesuaikan dengan format hasil solusi sistem. Hal tersebut untuk mendapatkan perbandingan performa diantara keduanya.

3.2.8. Penyusunan Buku Tugas Akhir

Tahap penyusunan buku tugas akhir merupakan proses pendokumentasian hasil tugas akhir serta analisis terhadap hasil akhir yang didapatkan. Keluaran dari tahap ini adalah buku tugas akhir. Buku tugas akhir ini diharapkan dapat menjadi referensi bagi penelitian selanjutnya.

BAB IV
PERANCANGAN

Dalam bab ini akan dijelaskan terkait persiapan perancangan yaitu perihal data yang dibutuhkan dan pembentukan model matematis dengan memperhatikan batasan yang sesuai dengan keadaan saat ini pada Departemen Sistem Informasi ITS.

4.1. Pengumpulan dan Deskripsi Data

Dalam pengerjaan tugas akhir ini, data yang digunakan merupakan data peserta yang mengikuti perkuliahan serta jadwal mata kuliah semester genap dan ganjil tahun ajaran 2017/2018. Selain itu terdapat pula data kurikulum yang berisi beban SKS untuk tiap mata kuliah. Data *features* dan juga ruang kelas yang digunakan akan menjadi masukan juga untuk membentuk dataset. Seluruh data diatas diperoleh dari Departemen Sistem Informasi, ITS.

4.1.1. Deskripsi Data Peserta Kuliah

Data peserta kuliah digunakan untuk mendapatkan informasi terkait mata kuliah yang diambil oleh masing masing mahasiswa aktif Departemen Sistem Informasi. Data tersebut terdiri dari nama mahasiswa, nrp, mata kuliah serta kelas yang diambil. Data peserta kuliah yang diberikan berupa file dengan format Microsoft Excel yang dibagi menjadi semester genap dan ganjil. Dari data peserta kuliah yang didapat, terdapat 651 mahasiswa pada semester ganjil dan 538 mahasiswa pada semester genap. Berikut merupakan gambaran data peserta kuliah semester ganjil pada Tabel 4.1dan semester genap pada Tabel 4.2

Tabel 4.1 Contoh data peserta kuliah semester ganjil

NO	NRP	NAMA	MK	KELAS
1	5211540000121	KHANSA AL-FAIZIY	ASD	A
2	5211640000021	ALIFIA INTAN DWI SAFITRI	ASD	A
3	5211640000031	JANIS RAMADHANTI SAPUTRI	ASD	A
4	5211640000049	CLARA GUSTIKA NANDA	ASD	A
5	5211640000059	MUHAMAD AINUR RIZAL	ASD	A
6	5211640000075	SYANANTA PUTRA RAMADHAN	ASD	A
7	5211640000076	BERRY HUMAIDI FUAD	ASD	A
8	5211640000077	AHMAD RIZQI FADLIL	ASD	A
9	5211640000090	ZENDIKA DAYONGKI SIPUTRI	ASD	A
10	5211640000099	MOH. FERIAN FAKHRUL ZAIN	ASD	A

Tabel 4.2 Contoh data peserta kuliah semester Genap

No	NRP	Nama	Kelas
1	5211640000132	R.DIMAS AGUNG WICAKSONO	ARSITI A
2	5211640000138	MUHAMAD RIZALDI SATRIO FADLI	ARSITI A
3	5211640000152	CHAMDANA TAQIE SAMBORO	ARSITI A
4	5211740000001	ADELLYA RIZQY DAMAYANTI	ARSITI A
5	5211740000002	MUHAMMAD ALFIAN SYAH	ARSITI A
6	5211740000003	A PAHMI	ARSITI A
7	5211740000005	NITA AMBARWATI	ARSITI A
8	5211740000018	RAINAL YUSRIL BAHRUNNAFAR	ARSITI A
9	5211740000019	NUR AINI LESTARI	ARSITI A
10	5211740000020	RETNO DWI KINASIH	ARSITI A

4.1.2. Deskripsi Data Jadwal Kuliah

Dalam pengerjaan tugas akhir ini, membutuhkan masukan berupa jadwal kuliah yang dibuat secara manual oleh pihak Departemen Sistem Informasi, ITS. Terdapat dua jadwal yaitu jadwal untuk semester ganjil dan genap tahun ajaran 2017/2018. Jadwal mata kuliah tersebut berisi informasi terkait hari, jam, mata kuliah, ruang kelas yang digunakan,

jumlah beban sks mata kuliah, posisi semester mata kuliah dan juga dosen yang mengampu mata kuliah tersebut. Jadwal untuk semester ganjil dan jadwal untuk semester genap dapat dilihat pada Lampiran B. Berdasarkan data jadwal mata kuliah pada Lampiran B terdapat beberapa informasi yang dapat diringkas. Tiap semester memiliki 13-14 timeslot/minggu yang bisa digunakan untuk penempatan mata kuliah. Terdapat 11 ruangan yang digunakan. Dan untuk mata kuliah yang diselenggarakan pada semester ganjil adalah 32 dan 36 mata kuliah pada semester genap.

4.1.3. Deskripsi Data *Features*, Ruang Kelas, dan Kapasitas Ruang Kelas

Informasi selanjutnya berkaitan dengan kebutuhan dataset yang nantinya diolah adalah data *features* dan ruang kelas. *Features* merupakan fasilitas apa saja yang akan digunakan untuk masing-masing mata kuliah yang akan dijadwalkan.Informasi tersebut akan digunakan untuk menyesuaikan ruang kelas yang sesuai dengan fasilitas yang dibutuhkan mata kuliah tertentu. Selain itu ada informasi untuk kapasitas ruangan yang digunakan berkaitan dengan jumlah mahasiswa yang akan dijadwalkan di kelas tersebut.Tabel 4.3 dan Tabel 4.4 dibawah ini menunjukkan daftar fasilitas dari setiap ruang yang ada serta kapasitas ruangan untuk masing-masing semester.

Tabel 4.3 Data ruang, features dan kapasitas ruang semester ganjil

Kelas	Features	Kapasitas
AULA	LCD+Proyektor	40
TC105A	LCD+Proyektor	40

Kelas	Features	Kapasitas
Studio Lt 2	LCD+Proyektor	45
TC101	LCD+Proyektor	45
TC103	LCD+Proyektor	45
TC104	LCD+Proyektor	45
TC106	LCD+Proyektor	45
TC105	LCD+Proyektor	50
TC107	LCD+Proyektor	50
TC108	LCD+Proyektor	50
Studio Lt 1	Komputer	45
Studio Lt 1	Software SAP	
Studio Lt 1	LCD+Proyektor	
Studio Lt 1	Software Eviews	
Studio Lt 1	Software Minitab	
Studio Lt 1	Software Excel	

Tabel 4.4 Data ruang, features dan kapasitas ruang semester genap

Kelas	Features	Kapasitas
AULA	LCD+Proyektor	40
TC105A	LCD+Proyektor	40
Studio Lt 2	LCD+Proyektor	45
TC101	LCD+Proyektor	45
TC103	LCD+Proyektor	45
TC104	LCD+Proyektor	45
TC106	LCD+Proyektor	45
TC105	LCD+Proyektor	50
TC107	LCD+Proyektor	50
TC108	LCD+Proyektor	50

Kelas	Features	Kapasitas
Studio Lt 1	Komputer	45
Studio Lt 1	Software SAP	
Studio Lt 1	LCD+Proyektor	
Studio Lt 1	Software Eviews	
Studio Lt 1	Software Minitab	
Studio Lt 1	Software Excel	

Berdasarkan informasi diatas dapat dilihat terdapat beberapa fitur yang dimiliki dan dibutuhkan oleh setiap ruangan. Total fitur yang dimiliki tiap semester terangkum dalam Tabel 4.5 dibawah ini

Tabel 4.5 Data Jumlah Fitur Tiap Semester

Semester	Jumlah Fitur
Ganjil 2017/2018	6 Fitur
Genap 2017/2018	5 Fitur

4.2. Pra Proses Data

Tahap pra proses data merupakan pengolahan data awal yang akan dijadikan file masukan untuk aplikasi penjadwalan mata kuliah otomatis. Data input yang dibuat mengikuti format dataset *Socha*. File tersebut terdiri dari beberapa komponen antara lain inisiasi data awal, kapasitas kelas dan beberapa matriks terkait informasi penjadwalan mata kuliah. Data yang dibuat dibedakan berdasarkan semester ganjil dan genap.

4.2.1. Inisiasi Data Awal

Baris pertama merupakan inisiasi data terkait informasi yang akan digunakan dalam penjadwalan mata kuliah otomatis.

Terdiri dari 4 kolom yang masing-masing merupakan angka yang memiliki arti sbegai berikut :

- Kolom 1 : jumlah event pada semester itu
- Kolom 2 : jumlah ruangan yang digunakan pada semester itu
- Kolom 3 : jumlah *features* yang digunakan pada semester itu
- Kolom 4 : jumlah mahasiswa yang mengambil mata kuliah pada semester itu

Tabel 2.1 merupakan contoh dari inisiasi data awal semester ganjil dan genap 2017/2018

Tabel 4.6 Inisiasi awal

Semester	Kolom 1	Kolom 2	Kolom 3	Kolom 4
Ganjil	118	11	6	651
Genap	106	11	5	538

4.2.2. Inisiasi Kapasitas Kelas

Baris selanjutnya menunjukkan kapasitas ruang kelas yang digunakan dalam perkuliahan semester genap dan ganjil. Baris kapasitas kelas mengikuti jumlah kelas yang sudah diinisiasi pada baris pertama kolom kedua. Gambar 4.1 menggambarkan baris kapasitas kelas untuk semester genap dan ganjil

2	40
3	40
4	45
5	45
6	45
7	45
8	45
9	50
10	50
11	50
12	45

Gambar 4.1 Data kapasitas ruang kelas

4.2.3. Matriks *Student* x *Event*

Baris setelah data kapasitas kelas merupakan matriks *student x event.Student* merupakan daftar mahasiswa yang mengambil mata kuliah pada semester itu. *Event* merupakan pertemuan antara mata kuliah, timeslot, dan dosen yang mengajar. Data jadwal mata kuliah yang ada pada Lampiran B akan menjadi masukan untuk penentuan *event* yang ada. Selanjutnya *event* yang masih dalam bentuk nama mata kuliah beserta kelas akan diubah menjadi kode dengan format urutan numerik 3 digit. Untuk lebih lengkapnya dapat dilihat pada lampiran B.

Data mata kuliah yang diambil oleh setiap mahasiswa tersebut disimpan dalam Microsoft Excel. Kemudian dengan bantuan *Pivot Table* maka dibuat matriks dengan nilai kolom sebagai kode *event* dan baris sebagai nrp mahasiswa. Nilai yang dikeluarkan berupa 1 atau 0, yang memiliki arti bahwa 1 akan muncul saat mahasiswa pada baris ke-*i* mengambil *event* pada kolom ke-*j*, sedangkan nilai 0 muncul pada kolom *event* ke-*j* yang tidak diambil mahasiswa baris ke-*i*. matriks yang

terbentuk sebesar jumlah *student* x *event.* Matriks 2 dimensi tersebut selanjutnya di transform menjadi matriks 1 dimensi untuk kemudian dimasukkan kedalam dataset tepat dibawah inisiasi kapasitas ruang kelas. Gambar 4.2 Menunjukkan sebagian matriks *student* x *event* yang telah di transform dan dimasukkan dalam dataset

13	0	207	0
14	0	208	0
15	0	209	0
16	0	210	0
17	0	211	0
18	0	212	0
19	0	213	0
20	0	214	0
21	0	215	0
22	0	216	1
23	0	217	0
24	0	218	0
25	0	219	0
26	0	220	0
27	0	221	1
28	0	222	0
		223	0
		224	0

Gambar 4.2 Dataset student x event

4.2.4. Matriks *Room* x *Features*

Matriks selanjutnya yang akan dibentuk merupakan matriks pertemuan antara *room* dan *features*. Dimana *room* akan menjadi baris dan *features* menjadi kolom.Matriks ini digunakan untuk mengetahui fitur apa saja yang dimiliki masing-masing ruangan. Nilai yang dikeluarkan berupa 1 atau 0, yang memiliki arti bahwa 1 akan muncul saat ruangan pada

baris ke-*i* memiliki fitur pada kolom ke-*j*, sedangkan nilai 0 muncul pada kolom fiturke-*j* yang tidak dimiliki ruangan ke-*i*.Matriks yang terbentuk sebesar jumlah *room*x *fetures.* Matriks 2 dimensi tersebut selanjutnya di transform menjadi matriks 1 dimensi untuk kemudian dimasukkan kedalam dataset tepat dibawah matriks *student* x *event*. Gambar 4.3 menunjukkan sebagian matriks *room* x *fetures* yang telah di transform dan dimasukkan dalam dataset

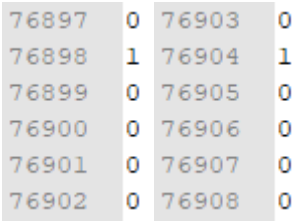
76831	0	76837	1
76832	1	76838	1
76833	0	76839	1
76834	0	76840	1
76835	0	76841	1
76836	0	76842	1

Gambar 4.3 Dataset room x features

4.2.5. Matriks *Event* x *Features*

Matriks terakhir yang dibuat untuk dataset adalah pertemuan antara data *event* dan juga data *features*. Matriks ini digunakan untuk membantu dalam nantinya mendapatkan informasi terkait dengan fitur yang dibutuhkan untuk setiap *event* yang dilaksanakan pada semester tersebut. Informasi lain yang dapat diambil terkait ruang apa saja yang bisa digunakan nantinya oleh masing-masing *event* dengan memperhatikan fitur yang dibutuhkan oleh *event*. Nilai yang dikeluarkan berupa 1 atau 0, yang memiliki arti bahwa 1 akan muncul saat *event* pada baris ke-*i* membutuhkan fitur pada kolom ke-*j*, sedangkan nilai 0 muncul pada kolom fiturke-*j* yang tidak dibutuhkan *event* ke-*i.* Matriks yang terbentuk sebesar jumlah *event*x *fetures.* Matriks 2 dimensi tersebut selanjutnya di

transform menjadi matriks 1 dimensi untuk kemudian dimasukkan kedalam dataset tepat dibawah matriks *event*x *features*. Gambar 4.4 menunjukkan sebagian matriks event x *fetures* yang telah di transform dan dimasukkan dalam dataset



76897	0	76903	0
76898	1	76904	1
76899	0	76905	0
76900	0	76906	0
76901	0	76907	0
76902	0	76908	0

Gambar 4.4 Dataset Event *x* features

4.2.6. Dataset TIM

Tahap praproses diakhiri dengan membuat gabungan dari 5 komponen yang sudah dijelaskan pada poin 4.2.1 hingga 4.2.5 diatas. Dataset diawali dengan inisiasi awal pada baris pertama, kemudian dilanjutkan dengan inisiasi kapasitas ruang kelas sepanjang jumlah kelas yang ada, lalu matriks *event* x *student*, matriks *room* x *features*, matriks *event* x *features*. File hasil gabungan tersebut lalu disimpan dalam satu format yaitu .tim seperti dataset *socha* yang nantinya akan menjadi input program penjadwalan mata kuliah otomatis.

4.3. Formulasi Model

Penentuan formulasi model dibutuhkan untuk memperjelas apa yang akan dikerjakan, mulai dari apa yang akan ditentukan, batasan-batasan yang harus dipatuhi, hingga fungsi tujuan yang harus dicapai oleh program yang dibuat.

4.3.1. Asumsi Notasi

Penjadwalan mata kuliah dilakukan dalam jangka waktu satu semester dengan jumlah timeslot sebanyak 13-14/minggu. Berikut notasi-notasi yang digunakan dalam penjadwalan otomatis :

i = Event yang dijadwalkan
t = Timeslot yang tersedia
r = Ruangan yang digunakan
c = Kapasitas ruangan

4.3.2. Variabel Keputusan

Berikut merupakan variable keputusan dari penelitian ini

$$X_{itr}\begin{cases} 1, \text{jika event } i \text{ dijadwalkan pada timeslot } t, \\ \quad \text{dan ruangan } r \\ \\ 0, \text{untuk kondisi lainnya} \end{cases} \quad (1)$$

Dengan :
X merupakan variabel keputusan
$i = \{1,2,3, \ldots . I\}$merupakan urutan *event*
$t = \{1,2,3, \ldots , T\}$ merupakan urutan timeslot pada jadwal
$r = \{1,2,3, \ldots , R\}$ merupakan urutan ruang kelas

4.3.3. Batasan Masalah

Dalam melakukan implementasi algoritma akan menggunakan beberapa batasan untuk ditepati yang dikelompokkan menjadi batasan keras (*hard constraint)* dan batasan lunak (*soft constraint*). Berikut merupakan *hard constraint* dan *soft consraint* yang digunakan

4.3.3.1. *Hard Constraint*

Hard constraint merupakan aturan yang wajib dipatuhi oleh model yang akan dibuat. Ketika sebuah solusi melanggar *hard constraint* yang ada, maka dinyatakan *not feasible*. *Hard constraint* yang pertama terkait dengan aturan bahwa setiap mata kuliah yang telah didefinisikan sebagai *event* harus terjadwalkan pada timeslot yang sudah ada. Dapat dilihat pada persamaan (2) dibawah ini

$$\sum_{\mathbf{i=1}}^{\mathbf{I}} \sum_{\mathbf{t=1}}^{\mathbf{T}} \sum_{\boldsymbol{r=1}}^{\boldsymbol{R}} \boldsymbol{X_{itr}} = \mathbf{1} \qquad (2)$$

Dimana X_{it} merupakan *event* ke-*i* yang dijadwalkan pada *timeslot* ke-*t.* Batasan selanjutnya adalah terkait bentrok jadwal. Setiap *event* yang sudah terjadwalkan tidak boleh memiliki timeslot yang sama satu sama lain, dapat dimodelkan seperti persamaan (3) dibawah ini

$$\sum_{i=1}^{I-1} \sum_{j=i+1}^{I} C_{ij}.V_{ij} = 0 \qquad (3)$$

$$V_{ij} \begin{cases} 1\ jika\ t_i = t_j \\ 0, untuk\ kondisi\ lainnya \end{cases}$$

Dengan :

C_{ij} = jumlah mahasiswa peserta *event* i dan j

V_{ij} = *vector* yang menunjukkan apakah mata kuliah i dan mata kuliah j bentrok

t_i = *timeslot* dimana *event* ke i dijadwalkan

Persamaan diatas menunjukkan bahwa timeslot untuk mata kuliah yang memiliki konflik tidak boleh dijadwalkan

bersamaan, konflik disini akan dijelaskan pada subab 4.3.5 berikutnya.

Yang ketiga adalah *hard constraint* terkait jumlah mahasiswa yang mengambil *event* tertentu tidak boleh melebihi kapasitas ruangan yang digunakan, dapat dimodelkan seperti persamaan (3) dibawah ini

$$\sum_{i=1}^{I} S_i X_{itr} \leq \sum_{r=1}^{R} P_r \qquad (3)$$

Dengan :
$i = \{1,2,3, \ldots . I\}$merupakan urutan *event*
$S_i =$ merupakan jumlah mahasiswa untuk *event* ke-i
$P_r =$ merupakan kapasitas ruang untuk ruang ke-r

Hard constraint yang terakhir adalah terkait bentrok yang melibatkan timeslot dan ruang. *Event* yang ada tidak boleh terjadwal dalam timeslot dan ruang yang sama, sehingga pada setiap *timeslot* yang dimiliki hanya dijadwalkan pada satu ruang. *Hard constraint* ini sama dengan persamaan (1) dimana *event i* yang dijadwalkan pada *timeslot* t dan ruang r maka akan diberi nilai 1.

4.3.3.2. *Soft Constraint*

Soft constraint merupakan batasan atau aturan yang akan lebih baik jika dipenuhi. Semakin sedikit *soft constraint* yang dilanggar, maka nilai pinalti yang didapat juga sedikit. Begitupun sebaliknya, semakin banyak *soft constraint* yang dilanggar, maka akan menambah nilai pinalti untuk tiap *soft constraint*-nya. Pada penelitian kali ini, terdapat 3 *soft*

constraint yang akan digunakan berdasarkan dengan keadaan Departemen Sistem Informasi.

Soft constraint pertama terkait aturan jumlah mata kuliah mahasiswa tiap harinya. Setiap mahasiswa tidak boleh mengambil lebih dari dua *event* pada hari yang sama. Dapat dilihat pada persamaan (4) dibawah ini

$$\sum_{s=1}^{S}\sum_{d=1}^{D} Y_{sd} \leq 2 \qquad (4)$$

Dengan :
$s = \{1,2,3, \ldots . S\}$merupakan urutan mahasiswa
$d = \{1,2,3, \ldots , D\}$ merupakan urutan hari pada jadwal
Y_{sd} = nilai mahasiswa ke *s* setiap hari ke *d*

Selanjutnya adalah *soft constraint* kedua, yaitu setiap mahasiswa tidak boleh hanya mengambil satu *event* dalam satu hari. Dapat dilihat pada persamaan (5) dibawah ini.

$$\sum_{s=1}^{S}\sum_{d=1}^{D} Y_{sd} > 1 \qquad (5)$$

Dengan :
$s = \{1,2,3, \ldots . S\}$merupakan urutan mahasiswa
$d = \{1,2,3, \ldots , D\}$ merupakan urutan hari pada jadwal
Y_{sd} = nilai mahasiswa ke *s* setiap hari ke *d*

Yang terakhir adalah *soft constraint* terkait pengambilan mata kuliah pada jam akhir. Mahasiswa tidak disarankan

mengambil mata kuliah yang terletak pada sesi terakhir setiap harinya. Dapat dilihat pada persamaan (6)

$$A_{si} = 0, \text{ Untuk t mod 11 = 0}$$

Dengan :
$s = \{1,2,3, \dots, o\}$ adalah urutan mahasiswa
$i = \{1,2,3, \dots, m\}$ adalah urutan event
$A_{si} =$ nilai mahasiwa ke s pada event ke i

4.3.4. Fungsi Tujuan

Fungsi tujuan yang digunakan dalam penjadwalan matakuliah dapat dilihat pada persamaan (7) dibawah ini :

$$Minimize\ P = P_1 + P_2 + P_3 \qquad (7)$$

$$P_1 = \sum_{s=1}^{S} \sum_{d=1}^{D} Y_{sd}$$

$$P_2 = \sum_{s=1}^{S} \sum_{d=1}^{D} Y_{sd}$$

$$P_3 = \sum_{s=1}^{S} \sum_{i=1}^{I} A_{si}$$

Dengan :
P_1 = akumulasi nilai pinalti mahasiswa yang mengambil < 2 *event* dalam satu hari
P_2 = akumulasi nilai pinalti mahasiswa mengambil hanya satu mata kuliah dalam satu hari
P_3 = akumulasi nilai pinalti mahasiswa mengambil mata kuliah pada jam terakhir

4.3.5. Conflict Matrix

Conflict matrix merupakan salah satu matriks yang nantinya akan sangat dibutuhkan dalam pengerjaan. *Conflict matrix* berukuran *event* x *event*. Matriks ini akan menunjukkan hubungan konflik untuk setiap eventnya. Sumbu y akan merepresentasikan *event* ke-i, sedangkan sumbu x akan merepresentasikan *event* ke-j. untuk setiap perpotongan *event* akan dilihat apakah terdapat mahasiswa yang sama mengambil *event* ke-i dan *event* ke-j sekaligus. Nilai yang akan mengisi matriks ini adalah ≤ 1 atau 0. Untuk menandakan bahwa terdapat mahasiswa yang sama mengambil *event* ke-i dan *event* ke-j sekaligus maka akan diberikan nilai 1 untuk tiap mahasiswa yang sama tersebut. Hal tersebut berarti *event* ke-i tersebut memiliki konflik dengan *event* ke-j sebanyak jumlah mahasiswa yang mengisi matriks konflik tersebut. Sedangkan 0 akan diisikan ketika tidak ada mahasiswa yang sama mengambil *event* ke-i dan *event* ke-j. Itu berarti tidak ada konflik diantara kedua *event*tersebut.

4.4. Algoritma Tabu-Variable Neighborhood Search Based Hyper Heuristic

Selanjutnya dari praproses data diatas akan menghasilkan solusi awal yang selanjutnya akan dioptimasi menggunakan algoritma *tabu-variable neighborhood search based hyper heuristic.* Algoritma tersebut antinya akan mencari solusi baru dengan berdasar pada solusi awal. Akan dilakukan pencarian solusi dengan melakukan teknik *swap* dan *move* untuk memodifikasi solusi awal. Setelah itu akan dihitung lagi nilai pinalti untuk setiap solusi baru yang dihasilkan. Selengkapnya akan dijelaskan pada bab V implementasi.

Halaman ini sengaja dikosongkan

BAB V
IMPLEMENTASI

Bab ini menjelaskan proses pelaksanaan penelitian tugas akhir dan proses implementasi algoritma *tabu – variable neigborhood search based hyperheuristics* dengan menggunakan bahasa pemrogaman java.

5.1. Lingkunngan Uji Coba

Pada subab Lingkungan Uji coba ini akan menjelaskan terkait lingkungan pengujian dalam melakukan implementasi penelitian tugas akhir terkait optimasi penjadwalan mata kuliah pada Departemen Sistem Informasi, Institut Teknologi Sepuluh Nopember. Spesifikasi perangkat keras yang digunakan dalam implementasi ditunjukkan pada Tabel 5.1

Tabel 5.1 Spesifikasi perangkat keras

Perangkat Keras	Spesifikasi
Jenis	Asus X200MA
Processor	Intel(R) Celeron(R) CPU 847
RAM	4.00 GB
Hard Disk Drive	1000 GB

Untuk spesifikasi perangkat lunak yang digunakan dalam implementasi metode ditunjukkan pada Tabel 5.2.

Tabel 5.2 Spesifikasi perangkat lunak

Perangkat Lunak	Fungsi
Windows 10 64 bit	Sistem Operasi
Microsoft Excel 2016	Pengolahan data

Perangkat Lunak	Fungsi
Netbeans 8.01	Implementasi algoritma
Microsoft Word 2016	Penulisan Laporan

5.2. Membaca File Input

Pada sub-bab ini akan menjelaskan proses awal yang harus dilakukan dalam implementasi. Hal tersebut adalah membaca file yang menjadi input untuk proses selanjutnya. Terdapat 2 file yang merupakan input dari penelitian tugas akhir ini. Yang pertama adalah file dengan ekstensi tim. Yang kedua merupakan file dengan ekstensi sol. Kedua file tersebut akan dibaca denga menggunakan bahasa pemrograman java.

5.2.1. Membaca file .TIM

File yang pertama merupakan file input yang berisi data set socha. File tersebut berisi informasi terkait ketetentuan mata kuliah pada semester tertentu. Terdapat informasi jumlah event, jumlah ruang yang digunakan, jumlah fitur, jumlah mahasiswa dan informasi yang lain seperti yang sudah dijelaskan pada bab 4.2 sebelumnya. Kode 5.1 berikut merupakan *source code* digunakan untuk membaca file input TIM

```
1. public void readTim(String filename) throws FileNotFoundException, IOException {
2. File file = new File(filename);
3. BufferedReader b = new BufferedReader(new FileReader(file));
4. String barissatu = b.readLine();
5. String[] listSatu = barissatu.split(" "); //baris 1
6. event = Integer.parseInt(listSatu[0]);
```

```
7.  room = Integer.parseInt(listSatu[1]);
8.  features = Integer.parseInt(listSatu[2]);
9.  student = Integer.parseInt(listSatu[3]);
10. roomData = new String[room];
11. dataRoom = new Integer[room];
12. for (int i = 0; i < room; i++) {
13.     roomData[i] = b.readLine();
14.     dataRoom[i] = Integer.parseInt(roomData[i]);
15. }
16. allCapacity = 0;
17. for (int num : dataRoom) {
18.     allCapacity = allCapacity + num;
19. }
```

Kode 5.1 Kode Program baca file .tim inisiasi awal

Pertama yang dilakukan adalah membuat *methodreadTIM* yang akan digunakan untuk membaca file dengan ekstensi TIM. Didalamnya terdapat beberapa fungsi diantaranya adalah mengambil file yang akan digunakan serta membacanya. Selanjutnya adalah membaca baris pertama file TIM lalu menjadikannya nilai *integer* agar mudah jika akan digunakan selanjutnya. Setelah membaca satu per satu nilai pada baris pertama, selanjutnya membaca nilai pada beberapa baris selanjutnya yang merupakan kapasitas ruang untuk masing-masing ruang dan memasukkannya dalam *array* satu dimensi. Lalu menghitung kapasitas total ruangan untuk setiap *timeslot* yang ada. Kode program selanjutnya akan digambarkan pada Kode 5.2 berikut

```
1.  /**
2.      * membuat matriks student x event
```

```
3.      */
4.    eventStudent = new Integer[student][event];
5.    for (int i = 0; i < student; i++) { //student
6.       for (int j = 0; j < event; j++) { // event
7.      eventStudent[i][j]=Integer.parseInt(b.readLine());
8.       }
9.    }
10.
11.  /**
12.   * membuat matriks room x features
13.   */
14.  roomFeatures = new Integer[room][features];
15.  for (int i = 0; i < room; i++) { //room
16.     for (int j = 0; j < features; j++) { //features
17.     roomFeatures[i][j]=Integer.parseInt(b.readLine());
18.     }
19.  }
20.
21.  /**
22.   * membuat matriks event x features
23.   */
24.  eventFeatures = new Integer[event][features];
25.  for (int i = 0; i < event; i++) { //event
26.       for (int j = 0; j < features; j++) { //features
27.    eventFeatures[i][j]=Integer.parseInt(b.readLine());
28.       }
29.  }
30.
31.  /**
32.   * membuat matriks jumlah student yang mengambil
     matkul i
```

```
33. */
34. studentNumber = new Integer[event];
35. Arrays.fill(studentNumber, 0);
36. for (int i = 0; i < event; i++) {
37.     for (int j = 0; j < student; j++) {
38.         studentNumber[i] = studentNumber[i] + eventStudent[j][i];
39.     }
40. }
```

Kode 5.2 Kode Program membuat matriks file TIM

Langkah selanjutnya adalah menterjemahkan data set pada baris selanjutnya menjadi matriks-matriks seperti sebelum di pra proses agar mudah digunakan dalam proses selanjutnya. Matriks-matriks tersebut akan disimpan dalam *array* 2 dimensi dengan panjang *i* dan *j* sesuai kebutuhan. Yang terakhir pada *method* ini adalah membuat *array* satu dimensi yang berisi jumlah mahasiswa yang mengambil masing-masing *event* yang ada. *Array* tersebut akan digunakan untuk keperluan implementasi selanjutnya.

5.2.2. Membaca file .SOL

File kedua yang menjadi input adalah file dengan ekstensi SOL. File tersebut berisi solusi penjadwalan mata kuliah. File terebut nantinya akan digunakan untuk melihat apakah jadwal yang dihasilkan sudah *feasible* atau belum. Selain itu, file ini juga akan menentukan nilai dari fungsi tujuan yaitu pinalti. Pada Kode 5.3 berikut merupakan *source code* yang digunakan

```
1. public void readSol(String filename) throws FileNotFoundException, IOException {
2. File filesol = new File(filename);
```

```
3. BufferedReader a = new BufferedReader(new
   FileReader(filesol));
4. Integer barispertama = Integer.parseInt(a.readLine());
5. jumlahevent = barispertama;
6. timeslot = 13;
7. String[] barissetelahsatu = new String[barispertama];
8. timeslotsaja = new Integer[barispertama];
9. ruangsaja = new Integer[barispertama];
```

Kode 5.3 Kode Program baca file .sol

Untuk membaca file SOL, langkah yang dilakukan sama seperti saat membaca file TIM. Dengan menggunakan *BufferedReader* untuk membaca file yang sudah diambil dengan inisiasi *filesol*. Setelah itu dilanjutkan dengan membaca per baris nilai yang ada. Baris pertama dibaca sebagai *jumlahevent*. Dikarenakan pada file SOL tidak ada informasi terkait jumlah *timeslot*, maka diperlukan inisiasi jumlah *timeslot* sebanyak yang diperlukan. Selanjutnya membaca baris selanjutnya yang berisi nilai *timeslot* dan ruang, lalu kemudian menyimpannya dalam *array* satu dimensi.

```
1. for (int i = 0; i < barispertama; i++) {
2.   barissetelahsatu[i] = a.readLine();
3.   String[] barissol = barissetelahsatu[i].split(","); //
4.   for (String barissol1 : barissol) {
5.     timeslotsaja[i] = Integer.parseInt(barissol[0]); //mengambil
   timeslot
6.     ruangsaja[i] = Integer.parseInt(barissol[1]); //mengambil
   ruang
7.   }
8. }
```

Kode 5.4 Kode Program memisahkan file SOL

Dalam implementasi selanjutnya contohnya untuk mengecek *hard constraint*, informasi nilai *timeslot* dan ruang perlu dipisah untuk lebih mempermudah. *Source code* pada Kode 5.4 diatas menunjukkan cara memishkan nilai *timeslot* dan ruang. Selanjutnya nilai tersebut masing-masing akan disimpan dalam *array* satu dimensi.

5.3. Pembuatan *Matrix*

Untuk dapat mempermudah proses implementasi, terdapat beberapa matriks yang dibutuhkan. Pada sub-bab ini akan menjelaskan terkait matriks yang dibutuhkan, serta program yang digunakan untuk menghasilkan matriks terkait.

5.3.1. Membuat *Conflict Matrix*

Matriks yang pertama adalah *conflict matrix*. Merupakan matriks *array*dua dimensi yang berukuran *event*x *event*. Matriks ini akan berisi informasi terkait konflik yang terjadi antara masing-masing *event* yang ada. Dilihat dari *event* yang diambil oleh masing-masing mahasiswa yang ada pada matriks *studentEventMatrix*. Untuk setiap mahasiswa, akan dilihat *event* apa saja yang diambil, lalu akan ditambahkan dalam *conflictMatrix* dan disimpan. Jumlah konflik antar *event* akan terus bertambah jika ditemukan mahasiswa yang mengambil beberapa *event* yang sama. Kode 5.5 merupakan kode program untuk membuat *conflict matrix.*

```
1.  public void makeConflictMatrix() {
2.      conflictMatrix = new Integer[event][event];
3.      for (Integer[] conflictMatrix1 : conflictMatrix) {
4.      Arrays.fill(conflictMatrix1, 0);
5.      }
```

```
6.      for (int i = 0; i < studentEventMatrix.size(); i++) {
        for (int j = 0; j < studentEventMatrix.get(i).size() - 1; j++)
    {
7.        for(intk=j + 1; k < studentEventMatrix.get(i).size(); k++)
    {
8.         Integer eventi = (studentEventMatrix.get(i).get(j));
9.         Integer eventj = (studentEventMatrix.get(i).get(k));
10.          conflictMatrix[eventi][eventj] += 1;
11. conflictMatrix[eventj][eventi] += 1;
12.    }
13.    }
14.  }
15. }
```

Kode 5.5 Kode program membuat *conflict matrix*

5.3.2. Membuat *Student Event Matrix*

Matriks selanjutnya merupakan matriks *list of array list* dua dimensi yang dinamis. Dimensi *i* menggambarkan mahasiswa yang ada, kemudian dimensi *j* menggambarkan*event* yang diambil oleh mahasiswa ke-*i*. Matriks ini dibuat untuk menghemat waktu saat membutuhkan informasi *event* yang diambil oleh tiap mahasiswa. Informasi diambil dari matriks *eventStudent* yang telah dibuat sebelumnya pada subab 5.2.1. Pada matriks iniakan mencari dari matriks *eventStudent* yang memiliki nilai 1, yang berarti *event* ke-*j* tersebut diambil oleh mahasiswa ke-*i*. Setelah mendapatkan nilai 1, kemudian diambilindeks *j* pada matriks *eventStudent* yang merupakan kode *event*. Setelah itu dimasukkan dalam *list of array list* yang sudah disiapkan yaitu *studentEventMatrix*. Kode program implementasi *studentEventMatrix*digambarkan pada Kode 5.6 berikut ini

```
1.  public void makeMatrix() {
```

```
2.      studentEventMatrix = new ArrayList<>(student);
3.      for (int i = 0; i < student; i++) {
4.        ArrayList<Integer> temp = new ArrayList<>();
5.        for (int j = 0; j < event; j++) {
6.          if (eventStudent[i][j] == 1) {
7.              temp.add(j);
8.          }
9.         }
10.  studentEventMatrix.add(temp);
11.     }
12. }
```

Kode 5.6 Kode Program membuat *Student Event Matrix*

5.3.3. Membuat *Conflict List Matrix*

Yang berikutnya adalah matriks *conflict list* yang berisi daftar konflik untuk tiap *event*. Matriks ini dibuat untuk memudahkan dalam proses pencarian dan juga mempersingkat waktu *looping* saat menjalankan kode program. Matriks ini merupakan matriks yang berisikan daftar kode *event* yang menjadi konflik untuk *event i*.

```
1.  int [][] conflictList(){
2.          daftarEdgesbaru = new int[event][];
3.            for (int i = 0; i < event; i++) {
4.              ArrayList<Integer> temp = new
    ArrayList<Integer>();
5.              for (int j = 0; j < conflictMatrix[i].length; j++) {
6.                int jadwalBisa = conflictMatrix[i][j];
7.                if (jadwalBisa >= 1 && i != j) {
8.                  temp.add(j);
9.                }
10.             }
11.             int[] arrayTemp = new int[temp.size()];
```

```
12.                 for (int j = 0; j < temp.size(); j++) {
13.                     arrayTemp[j] = temp.get(j);
14.                 }
15.                 daftarEdgesbaru[i] = arrayTemp;
16.             }
17.             return daftarEdgesbaru;
18.     }
```

Kode 5.7 Kode program membuat *Conflict List Matrix*

Kode 5.7 diatas merupakan tahap pembuatan *conflict list matrix*. Matriks ini nantinya akan berisi ringkasan dari *conflict* matriks. Pertama yang dilakukan adalah menyimpan indeks *j* yang memiliki nilai ≤ 1 dan indeknya $\neq$ indeks *i*. Selanjutnya indeks *j* tersebut akan disimpan dalam *array* dua dimensi.

5.3.4. Membuat *Suitable Room Matrix*

Matriks *Suitable Room* merupakan matriks dua dimensi dengan ukuran *i* sebanyak *event* dan *j* sebanyak room. Matriks ini dibuat dengan memanfaatkan beberapa matriks yang telah dibuat sebelumnya, diantaranya matriks *roomFeature, eventFeatures*dan *studentNumber*.Yang pertama dilakukan adalah menggunakan 2 fungsi for dimana *i* akan berjalan sepanjang *event* dan *j* akan berjalan sepanjang *room*. Didalamnya terdapat kondisi dimana jika kapasitas untuk setiap ruangan$\geq$ jumlah mahasiswa yang mengambil *eventI,* maka akan menuju proses selanjutnya yaitu mengecek kesesuaian antar *event* dan *room*. Kesuaian tersebut dilihat dengan menggabungkan 2 matriks yaitu *roomFeature*dan *eventFeatures,* kemudian menjadikannya *suitableRoom*. Kode 5.8 dibawah ini menunjukkan kode program untuk implementasi matriks *suitableRoom.* Output dari kode program tersebut adalah matriks dengan nilai 1 atau 0, dimana

1 berarti bahwa *event i* tersebut bisa ditempatkan pada *room j*. Untuk nilai 0 berlaku sebaliknya.

```
1.  public void makeSuitableRoom(){
2.    suitableRoom = new Integer [event][room];
3.    for (int i = 0; i < event; i++) {
4.       for (int j = 0; j < room; j++) {
5.          if (dataRoom [j] >= studentNumber [i]) {
6.             for (int k = 0; k < features; k++) {
7.                if (eventFeatures [i][k] == 1 && roomFeatures
    [j][k] == 0) {
8.                suitableRoom[i][j] = 0;
9.                }
10.               else if (eventFeatures [i][k] == 1 && roomFeatures
    [j][k] == 1 ) {
11.               suitableRoom[i][j] = 1;
12.               }
13.            }
14.         }
15.         else{
16.            suitableRoom[i][j] = 0;
17.         }
18.      }
19.   }
20.   }
```

Kode 5.8 Kode program untuk membuat Suitable Room Matrix

5.3.5. Membuat *Conflict Timeslot Matrix*

Matriks terakhir yang dibuat berisi konflik *event* yang dijadwalkan pada *timeslot* yang sama. Matriks tersebut disimpan dalam*list of array list*. Matriks ini akan mencari nilai yang sama pada variabel timeslotsaja, kemudian mengambil indeks *event* yang memiliki *timeslot* tersebut. Setelah itu

dimasukkan dalam daftarConflictTs. Kode program yang digunakan dapat dilihat pada Kode 5.9 dibawah ini

```
1.  public void makeConflictTimeslot(){
2.  daftarConflictTs = new ArrayList<>(timeslot);
3.  for (int i = 0; i < timeslot; i++) {
4.  ArrayList<Integer> temp = new ArrayList<>();
5.  for (int j = 0; j < timeslotsaja.length; j++) {
6.  if (i+1 == timeslotsaja[j]) {
7.  temp.add(j);
8.  }
9.  }
10. daftarConflictTs.add(temp);
11. }
12. }
```

Kode 5.9 Kode Program membuat *Conflict Timeslot Matrix*

5.4. Implementasi Constraint

Pada tahap ini, akan dilakukan pemrograman untuk implementasi batasan yang sudah didefinisikan sebelumnya. Hal tersebut dilakukan agar batasan yang ada dapat diproses sesuai dengan input yang ada dan juga martriks yang telah dibuat.

5.4.1. Implementasi *Hard Constraint*

Hard constraint digunakan untuk menunjukkan fisibilitas dari solusi yang dihasilkan. Jika solusi melanggar *hard constraint* yang ada, maka akan dinyatakan tidak *feasible*. Sebaliknya jika memenuhi *hard constraint* maka akan dinyatakan *feasible.* Terdapat empat *hard constraint* yang akan digunakan dalam penelitian tugas akhir ini.

*Hard constraint*yang pertama adalah semua *event* yang ada harus terjadwalkan seluruhnya. Dengan menggunakan fungsi for, dilakukan pengecekan pada *array* timeslotsaja yang berisi *timeslot* untuk tiap *event*. Didalamnya terdapat kondisi ketika *timeslot event-i* bernilai 0, maka dinyatakan *not feasible*. Sebaliknya jika memiliki nilai ≥ 1, maka dinyatakan *feasible*. Kode program untuk *hard constraint* 1 dapat dilihat pada Kode 5.10 dibawah ini

```
1.  fisibilitas1 = new  Boolean [timeslotsaja.length];
2.          for (int i = 0; i < timeslotsaja.length; i++) {
3.              fisibilitas1 [i] = (timeslotsaja[i] != 0);
4.          }
```

Kode 5.10 Kode Program *Hard Constraint* 1

Selanjutnya *hard constraint* kedua adalah *event* yang memiliki konflik tidak boleh dijadwalkan dalam 1 timeslot yang sama. *Hard constraint* ini menggunakan bantuan matriks *conflictMatrix* untuk mengecek ada atau tidaknya konflik antar event. Saat menemukan nilai dalam *conflictMatrix*> 0 maka akan diambil indeks *i* dan *j*, untuk kemudian di cek *timeslot*nya pada *array* timeslotsaja. Kondisi *feasible* berlaku saat *timeslotevent* yang memiliki konflik dijadwalkan berbeda. Sebaliknya berlaku *not feasible* saat *event* yang memiliki konflik dijadwalkan pada *timeslot* yang sama. Kode program untuk *hard constraint* 2 dapat dilihat pada Kode 5.11 dibawah ini

```
1.  fisibilitas2 = new Boolean [event];
2.          for (int i = 0; i < event; i++) {
3.              for (int j = 0; j < event; j++) {
```

```
4.                 if (conflictMatrix[i][j] > 0) {
5.   fisibilitas2 [i] = (timeslotsaja[i].equals(timeslotsaja[j]));
6.                 }
7.             }
8.         }
```

Kode 5.11 Kode Program *Hard Constraint* 2

Hard constraint yang ketiga adalah jumlah mahasiswa yang mengambil *eventi* tidak boleh melebihi kapasitas dimana ruang *event i* tersebut dijadwalan.Fungsi *for* akan berjalan sebanyak jumlahevent yang ada. Kemudian akan mengecek apakah*array studentNumber* indeks *i*memiliki jumlah ≤ kapasitas ruangan dimana *event i* dijadwalkan. Saat jumlah mahasiswa yang mengambil *event* tersebut lebih banyak jika dibandingkan kapasitas ruangan, maka akan bernilai *not feasible*. Kode program *hard constraint* 3 dapat dilihat pada Kode 5.12 dibawah ini

```
1.  Boolean [] fisibilitas3 = new Boolean [jumlahevent];
2.          for (int i = 0; i < jumlahevent; i++) {
3.  fisibilitas3      [i]      =      (studentNumber[i]      <=
    Integer.parseInt(roomData[ruangsaja[i] - 1]));
4.          }
```

Kode 5.12 Kode Program *Hard Constraint* 3

Hard constraint terakhir adalah *timeslot* dan ruang yang sudah digunakan oleh *event* tertentu tidak boleh digunakan oleh *event* lain. Untuk memudahkan dalam pengecekan, maka terlebih dahulu membuat matriks dua dimensi berukuran *timeslot* x *room*. Matriks tr ini akanmenyimpan nilai 1 untuk setiap*timeslot i* dan room *j* sudah terpakai. Setelah terbentuk matriks tr, maka akan dicek nilai dalam matriks. Matriks yang

bernilai > 1 maka dinyatakan *not feasible*. Kode program *hard constraint* 4 dijelaskan pada Kode 5.13 dibawah ini

```
1.  fisibilitas4 = new Boolean [timeslot][room];
2.        tr = new Integer[timeslot][room];
3.        for (int i = 0; i < jumlahevent; i++) {
4.          int r = ruangsaja[i] - 1;
5.          int t = timeslotsaja[i] - 1;
6.          tr[t][r]++;
7.        }
8.        for (int i = 0; i < timeslot; i++) {
9.           for (int j = 0; j < room; j++) {
10.             fisibilitas4 [i][j] = (tr[i][j] <= 1);
11.          }
12.       }
13.   }
```

Kode 5.13 Kode Program *Hard Constraint* 4

5.4.2. Implementasi *Soft Constraint*

Tahap selanjutnya merupakan implementasi formulasi model dari *soft constraint*. Sama seperti halnya implementasi *hard constraint*, tahap ini juga bertujuan untuk memudahkan dalam penerapan algoritma selanjutnya. Selain itu, *soft constraint* juga digunakan untuk nantinya menghitung fungsi tujuan yang telah dibuat yaitu total pinalti yang dihasilkan dari setiap pelanggaran pada *soft constraint*. Oleh karena itu *soft costraint* harus pula diimplementasikan menggunakan java. Beberapa *soft constraint* yang digunakan akan dijelaskan berikut ini.

Sebelum masuk dalam kode program *soft constraint*, terlebih dahulu harus membuat matriks yang dibutuhkan. Yang pertama merupakan matriks *array of array list* yang berisi

timeslot berapa saja yang diambil oleh mahasiswa bersangkutan. Menggunakan fungsi *for*, kode program akan berjalan sebanyak mahasiswa yang ada. Membaca matriks *studentEventMatrix* untuk melihat *event* yang diambil oleh mahasiswa. Selanjutnya dengan menggunakan *array* timeslotsaja yang berisi informasi *timeslot* yang dijadwalkan untuk setiap *event*, *timeslot* tersebut akan diambil dan dimasukkan dalam *studentTimeslot* dalam bentuk *double.* Kode progam seperti pada Kode 5.14 dibawah ini

```
1.  public void cekSoftConstraint(){
2.     studentTimeslot = new ArrayList<List<Double>>(student);
3.     for (int i = 0; i < studentEventMatrix.size(); i++) {
4.        ArrayList<Double> temp = new ArrayList<Double>();
5.        for (int j = 0; j < studentEventMatrix.get(i).size(); j++) {
6.  temp.add((double)timeslotsaja[studentEventMatrix.get(i).get(j)
    ]);
7.        }
8.        Collections.sort(temp);
9.        studentTimeslot.add(temp);
10.    }  }
```

Kode 5.14 Kode Program Inisiasi matriks *student timeslot*

Selanjutnya untuk dapat memproses *soft constraint* pertama dan kedua, maka *timeslot* yang sudah ada harus dikelompokkan berdasarkan hari pelaksanaannya. Dalam pembahasan pada bab sebelumnya, diketahui bahwa *timeslot* per hari adalah sebanyak tiga, kecuali hari jumat hanya satu. Oleh sebab itu untuk dapat mengelompokkan *timeslot* yang ada menjadi per hari, maka dapat dilakukan dengan rumus $\frac{studentTimeslot(i)}{3}$, kemudian hasil pembagian tersebut

dibulatkan keatas. Dalam java, fungsi pembulatan tersebut menggunakan *math.ceil*. Kode 5.15 merupakan kode program pengelompokkan *timeslot*

```
1.  List<List<Double>> TimeslotDay = new
    ArrayList<List<Double>>(student);
2.      double tday = 3;
3.      for (int i = 0; i < studentTimeslot.size(); i++) {
4.          ArrayList<Double> temp = new
    ArrayList<Double>();
5.          for (double x : studentTimeslot.get(i)) {
6.            temp.add(Math.ceil(x / tday));
7.          }
8.          TimeslotDay.add(temp);
9.      }
```

Kode 5.15 Kode Program *Convert timeslot* menjadi per hari

Setelah membuat dua *array list* sebelumnya, lalu mulai melakukan implementasi *soft constraint.Soft contraint* yang pertama adalah setiap mahasiswa tidak disarankan mengambil > 2 *event* dalam satu hari. Implementasi *soft constraint* ini dapat digabungkan dengan *soft constraint* kedua yaitu setiap mahasiswa tidak disarankan mengambil hanya satu *event* dalam sehari. Dengan menggunakan fungsi *frequency*, akan dilihat untuk masing-masing *timeslot* yang diambil mahasiswa apakah hanya satu atau melebihi dua dalam sehari. Jika masuk dalam salah satu atau dua kondisi tersebut sekaligus, maka akan diberikan nilai pinalti sebesar 1 untuk setiap pelanggaran yang dilakukan. Kode program implementasi *soft constraint* 1 dan 2 dapat dilihat pada Kode 5.16.

```
1.  int sc12 = 0;
```

```
2.          for (int i = 0; i < TimeslotDay.size(); i++) {
3.              for (int j = 0; j < 6; j++) {
4.                  int freq = Collections.frequency(TimeslotDay.get(i),
    (double) j);
5.                  if (freq > 2 || freq == 1) {
6.                      sc12++;
7.                  }
8.              }
9.          }
```

Kode 5.16 Implementasi *Soft Constraint*(1) dan (2)

Yang terakhir adalah implementasi *soft constraint* ketiga yaitu, sebisa mungkin event tidak dijadwalkan pada timeslot terakhir setiap harinya. Implementasi *soft constraint* tersebut dilakukan dengan menggunakan *arrayliststudentTimeslot*. Dikarenakan jumlah *timeslot* per hari adalah 3, maka untuk mengetahui bahwa *timeslot* tersebut merupakan *timeslot* terakhir dapat dilakukan dengan cara modulo. Ketika dilakukan modulo 3 dan mendapatkan nilai 0, maka berarti *timeslot* tersebut merupakan *timeslot* terakhir pada hari tersebut. Kemudian akan diberikan nilai pinalti 1 untuk setiap *event* yang dijadwalkan pada *timeslot* tersebut. Kode program *soft constraint* dapat dilihat pada Kode 5.17.

```
1.  int sc3 = 0;
2.  for (int i = 0; i < studentTimeslot.size(); i++) {
3.      for (double x : studentTimeslot.get(i)) {
4.          if ((x % (double) 3) == (double) 0) {
5.              sc3++;
6.          }
7.      }
8.  }
```

```
9.  int fSol = sc12 + sc3;
```

Kode 5.17 Implementasi *Soft Constraint* (3)

Nilai pinalti yang sudah didapatkan dari masing-masing soft constraint selanjutnya akan dijumlahkan untuk mendapatkan fungsi tujuan yang sudah didefinisikan sebelumnya.

5.5. *Generate* SOL

Pembentukan solusi awal yang akan digunakan sebagai input pada proses optimasi dilakukan dengan menggungakan algoritma *greedy*. Solusi awal tersebut berisi *timeslot* dan ruang yang dialokasikan untuk setiap *event* yang ada. Dalam implementasi algoritma ini, dibutuhkan beberapa informasi awal seperti daftar konflik untuk setiap *event*, jumlah timeslot, jumlah mahasiswa tiap *event* serta kapasitas ruang tiap *timeslot*. Langkah awal adalah membentuk method okToSlot yang digunakan untuk memastikan apakah *event* yang bersangkutan boleh dijadwalkan pada *timeslot* tertentu dengan cara mengecek adakah bentrok dengan *event* sebelumnya yang sudah dijadwalkan. Kode program okToSlot dapat dilihat seperti Kode 5.18 dibawah ini

```
1.  boolean okToSlot(int lecture, int slot, int[][] schd) {
2.  for (int i = 1; i < schd[lecture].length; i++) {
3.  int ithAdjLecture = schd[lecture][i];
4.  if (subjectTimeSlot[ithAdjLecture] == slot) {
5.  return false;
6.  }
7.  }
8.  return true;
9.  }
```

Kode 5.18 Kode program penjadwalan *timeslot*

Kedua membuat method okToRoom yang digunakan untuk melakukan pengecekan ketersediaan sisa kapasitas ruang untuk setiap *timeslot.* Jika masih terdapat sisa kapasitas yang dapat menampung suatu *event*, maka *event* tersebut akan dijadwalkan dalam timeslot tersebut dan sisa kapasitas akan berkurang sebesar jumlah peserta yang mengambil *event* tersebut. Merupakan kode program untuk okToRoom

```
1. boolean okToRoom(int index, int session, int[] AmountStudentPerSubject) {
2. int a = AmountStudentPerSubject[index];
3. if (a <= remaincapacity[session - 1]) {
4. remaincapacity[session - 1] = remaincapacity[session - 1] - a;
5. return true;
6. }
7. return false;
8. }
```

Kode 5.19 Kode program mengecek sisa kapasitas ruang

Kedua method diatas harus dipenuhi oleh suatu *event* agar dapat dijadwalkan pada suatu timeslot. Ketika suatu *event* tidak memenuhi salah satu syarat diatas, maka akan diulang dengan mencoba pada timeslot berikutnya. Untuk dapat dijadwalkan, menggunakan kode program seperti Kode 5.20 dibawah ini

```
1. void explore(int[][] schd, int [] AmountStudentPerSubject){
2. for (int lecture = 0; lecture < schd.length; lecture++) {
3. int realMK = MK.get(lecture).getIndeks();
```

```
4.  for (int i = 1; i <= remaincapacity.length; i++) {
5.  if (
6.  okToSlot(realMK, i, schd) &&
7.  okToRoom(realMK, i, AmountStudentPerSubject)
8.  ){
9.  subjectTimeSlot[realMK] = i;
10. break;
11. }}}}
```

Kode 5.20 Kode program *generate sol*

5.6. Implementasi Algoritma *Tabu-Variable Neighborhood Search based Hyperheuristic*

Sub bab ini akan membahas implementasi *tabu- variable neighborhood serch based Hyperheuristics*pada solusi awal yang sudah dihasilkan sebelumnya.

5.6.1. Implementasi Algoritma *Variable Neighborhood Search*

Pada sub-sub bab ini akan menjelaskan implementasi algoritma *Variable Neighborhood Search* yang digunakan sebagai awal proses optimasi. Algoritma VNS tersebut memiliki strategi menggunakan lebih dari satu struktur lingkungan dan mengubah struktur tersebut secara sistematis selama pencarian lokal.Implementasi algoritma pada penelitian ini menggunakan strategi *swap* dan *move* untuk mendapatkan solusi barudari *local search*. Berikut penjelasan lebih lanjut terkait strategi *swap* dan move:

a. *Move*

   Strategi *move*digunakan untuk mengganti *timeslot* dengan yang baru. Pergantian tersebut dilakukan secara random baik indeks yang akan diganti maupun bilangan yang digunakan untuk mengganti. Strategi

tersebut bertujuan dapat meminimalkan nilai pinalti sehingga jadwal yang dihasilkan memiliki tingkat keadilan antar mahasiswa.

Terdapat beberapa struktur move yang digunakan dalam penelitian ini diantaranya :

- Memindahkan 1 *event* saat iterasi

  Pada struktur move yang pertama bekerja dengan cara memindahkan timeslot 1 *event*secara acak seperti dijelaskan pada Kode 5.21 dibawah ini

```
1.  void move1() {
2.  for (int i = 0; i < Iteration; i++) {
3.  this.jadwalBaru = jadwalAwal.clone();
4.  int randomSlot;
5.  randomLecture = (int) (Math.random() * timeslotsaja.length);
6.  do {
7.  randomSlot = (int) ((Math.random() * (timeslot) +1));
8.  }
9.  while (randomSlot == this.jadwalAwal[randomLecture]);
10. if (tsokToSlot(randomLecture, randomSlot, conflictList)){
11. this.jadwalBaru[randomLecture] = randomSlot;
12. hitung();
13. }}}
```

Kode 5.21 Kode Program Move1

- Memindahkan 2 *event* sekaligus saat iterasi

  Pada struktur move yang kedua bekerja dengan cara memindahkan 2 timeslot pada 2 *event*sekaligus secara acak seperti pada Kode 5.22 dibawah ini

```
1.  void move2() {
2.  for (int i = 0; i < Iteration; i++) {
3.  this.jadwalBaru = jadwalAwal.clone();
```

```
4.  int randomSlot;
5.  randomLecture = (int) (Math.random() * timeslotsaja.length);
6.  int randomLecture2 = (int) (Math.random() *
    timeslotsaja.length);
7.  do {
8.  randomSlot = (int) ((Math.random() * (timeslot) +1));
9.  randomSlot2 = (int) ((Math.random() * (timeslot) +1));
10. }
11. while (randomSlot == this.jadwalAwal[randomLecture]);
12. if (tsokToSlot(randomLecture, randomSlot, conflictList)&&
13. tsokToSlot(randomLecture2, randomSlot2, conflictList)){
14. this.jadwalBaru[randomLecture] = randomSlot;
15. this.jadwalBaru[randomLecture2] = randomSlot2;
16. hitung();
17. }}}
```

Kode 5.22 Kode Program Move2

- Memindahkan 3 *event* sekaligus saat iterasi
  Yang terkahir pada stuktur move adalah memindahkan *timeslot* untuk 3 *event* sekaligus secara random seperti Kode 5.23

```
1.  void move3() {
2.  for (int i = 0; i < Iteration; i++) {
3.  this.jadwalBaru = jadwalAwal.clone();
4.  int randomSlot;
5.  randomLecture = (int) (Math.random() * timeslotsaja.length);
6.  int randomLecture2 = (int) (Math.random() *
    timeslotsaja.length);
7.  int randomLecture3 = (int) (Math.random() *
    timeslotsaja.length);
8.  do {
9.  randomSlot = (int) ((Math.random() * (timeslot) +1));
```

```
10. randomSlot2 = (int) ((Math.random() * (timeslot) +1));
11. randomSlot3 = (int) ((Math.random() * (timeslot) +1));
12. }
13. while (randomSlot == this.jadwalAwal[randomLecture]);
14. if (tsokToSlot(randomLecture, randomSlot, conflictList)&&
15. tsokToSlot(randomLecture2, randomSlot2, conflictList)&&
16. tsokToSlot(randomLecture3, randomSlot3, conflictList)){
17. this.jadwalBaru[randomLecture] = randomSlot;
18. this.jadwalBaru[randomLecture2] = randomSlot2;
19. this.jadwalBaru[randomLecture3] = randomSlot3;
20. hitung();
21. }}}
```

Kode 5.23 Kode Program Move3

b. *Swap*

Strategi *swap* digunakan untuk menukar antara 2 *timeslot event*. Penukaran *timeslot* menggunakan cara *random* dengan tetap memperhatikan *hard constraint*. Solusi baru akan diterima saat dinyatakan *feasible* dengan memenuhi semua *hard constraint.*Terdapat 2 struktur swap yang digunakan :

- Menukar 2 *event*

  Struktur keempat yang digunakan adalah *swap* yaitu menukarkan 2 *timelot* yang dimiliki 2 *event* secara acak seperti Kode 5.24

```
1. void swap1() {
2.         for (int i = 0; i < Iteration; i++) {
3.             int swRandomLecture;
4.             this.jadwalBaru = jadwalAwal.clone();
5.             swRandomLecture = (int) (Math.random() *
   timeslotsaja.length);
```

```
6.          int swRandomLecture2 = (int) (Math.random()
    * timeslotsaja.length);
7.          while (swRandomLecture ==
    swRandomLecture2) {
8.              swRandomLecture2 = (int) (Math.random() *
    timeslotsaja.length);
9.          }
10.         if (
11.             tsokToSlot(randomLecture,
    jadwalBaru[swRandomLecture2], conflictList) &&
12.
    tsokToSlot(jadwalBaru[swRandomLecture2],
    jadwalBaru[swRandomLecture], conflictList) ){
13.             int temp = jadwalBaru[swRandomLecture];
14.             jadwalBaru[swRandomLecture] =
    jadwalBaru[swRandomLecture2];
15.             jadwalBaru[swRandomLecture2] = temp;
16.             hitung();
17.         }}}
```

Kode 5.24 Kode Program Swap1

- Menukar 4 *event* sekaligus
  Struktur lingkungan yang terakhir adalah menukarkan *timeslot* untuk 4 *event* sekaligus seperti Kode 5.25

```
1.  void swap2() {
2.  for (int i = 0; i < Iteration; i++) {
3.  int swRandomLecture, swRandomLecture2,
    swRandomLecture3, swRandomLecture4;
4.  this.jadwalBaru = jadwalAwal.clone();
5.  swRandomLecture = (int) (Math.random() *
```

```
    timeslotsaja.length);
6.  swRandomLecture2 = (int) (Math.random() *
    timeslotsaja.length);
7.  swRandomLecture3 = (int) (Math.random() *
    timeslotsaja.length);
8.  swRandomLecture4 = (int) (Math.random() *
    timeslotsaja.length);
9.  while (swRandomLecture == swRandomLecture2) {
    swRandomLecture2 = (int) (Math.random() *
    timeslotsaja.length);}
10. while (swRandomLecture == swRandomLecture3) {
    swRandomLecture3 = (int) (Math.random() *
    timeslotsaja.length);}
11. while (swRandomLecture == swRandomLecture4) {
    swRandomLecture4 = (int) (Math.random() *
    timeslotsaja.length);}
12. while (swRandomLecture2 == swRandomLecture3) {
    swRandomLecture3 = (int) (Math.random() *
    timeslotsaja.length);}
13. while (swRandomLecture2 == swRandomLecture4) {
    swRandomLecture4 = (int) (Math.random() *
    timeslotsaja.length);}
14. while (swRandomLecture3 == swRandomLecture4) {
    swRandomLecture4 = (int) (Math.random() *
    timeslotsaja.length);}
15. if (
    tsokToSlot(randomLecture, jadwalBaru[swRandomLecture2],
    conflictList) &&
    tsokToSlot(jadwalBaru[swRandomLecture2],
    jadwalBaru[swRandomLecture3], conflictList) &&
    tsokToSlot(jadwalBaru[swRandomLecture3],
    jadwalBaru[swRandomLecture4], conflictList) &&
    tsokToSlot(jadwalBaru[swRandomLecture4],
```

```
    jadwalBaru[swRandomLecture], conflictList)){
16. int temp = jadwalBaru[swRandomLecture];
17. jadwalBaru[swRandomLecture] =
    jadwalBaru[swRandomLecture2];
18. jadwalBaru[swRandomLecture2] =
    jadwalBaru[swRandomLecture3];
19. jadwalBaru[swRandomLecture3] =
    jadwalBaru[swRandomLecture4];
20. jadwalBaru[swRandomLecture4] = temp;
21. hitung();
22. }}}
```

Kode 5.25 Kode Program *Swap2*

5.6.2. Pemilihan Solusi dengan Menggunakan Tabu Search

Pada sub-sub bab ini akan dijelaskan implementasi algoritma *Tabu search* yang digunakan untuk memilih solusi terbaik yang dihasilkan saat optimasi.Dengan menggunakan algoritma *Tabu Search* akan ada *tabu list* yang dapat membantu dalam menyimpan strategi yang menghasilkan nilai pinalti lebih besar, sehingga tidak akan digunakan terlebih dahulu.Pseudocode*tabu search*yang digabung dengan algoritma VNS dapat dilihat pada Kode 5.26 dibawah ini

```
Initiation :
    1. Select neighborhood structure n_k, k = {1,2,3,….,K}:
       for i=0 to size k
    2. [string][] sol ← string[][]
    3. Set End
    4. [string][] bestSol ← string[][]
Repeat :
    1. for i = 0 to size k
       a. shaking (generate random solution sol from n_k
```

```
b. local search : random.math
c. move or not
   if ((f(bestSol) < s) or (f(bestSol) is accepted by the
   acceptance criterion)) then
      sol ← bestSol;
      set k ← 1;
      while k is in the tabulist and k < K
      k ← k+1;
      continue the search with n_k;
   else
      insert k to the tabulist;
      set k ← k+1;
      increase the tabu length by 1;
      if tabu length > tabu tenure
         release the first neighbourhood structure
   from the tabu list;
      while k is in the tabulist and k < K
   k ← k+1;
```

Kode 5.26 Pseudocode VNS-Tabu Hyperheuristic

BAB VI
HASIL DAN PEMBAHASAN

Pada bab ini akan menjelaskan mengenai proses uji coba dan juga hasil uji coba serta analisis terhadap hasil yang diperoleh dari proses implementasi Algoritma *Tabu-Variable Neighborhood Search Based Hyper-heuristics*.

6.1. Validasi *Generate SOL*

Validasi *generate SOL* dilakukan untuk memastikan kondisi solusi awal jadwal terhadap *hard constraint* yang ada. Proses validasi digunakan untuk memastikan kebenaran kode program yang telah dibangun. Hasil *generating* jadwal yang feasible terlampir pada lampiran C. Tabel 6.1 berikut merupakan hasil validasi *generate SOL* untuk semester ganjil dan genap.

Tabel 6.1 Pengecekan Solusi

Kode Hard Constraint	Solusi Awal	
	Ganjil	Genap
HC 1	Terpenuhi	Terpenuhi
HC 2	Terpenuhi	Terpenuhi
HC 3	Terpenuhi	Terpenuhi
HC 4	Terpenuhi	Terpenuhi

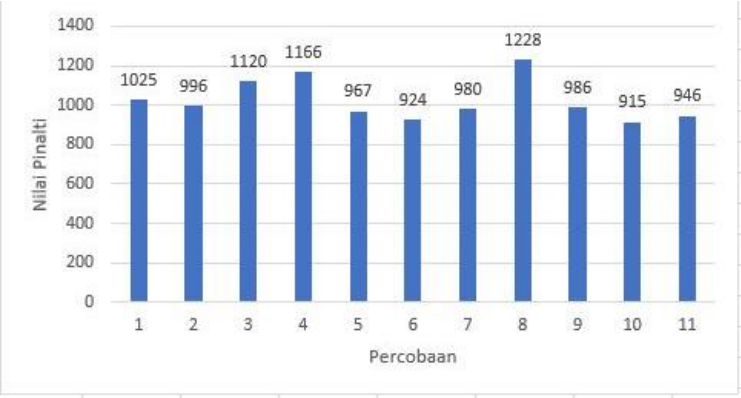
6.2. Hasil Implementasi Penjadwalan Semester Ganjil

Hasil implementasi pada semester ganjil akan dibandingkan menggunakan variabel nilai pinalti yang didapat untuk masing masing solusi yang dihasilkan. Nilai pinalti menunjukkan seberapa banyak solusi tersebut dapat memenuhi kriteria *soft constraint* yang ada. Skenario yang dibuat mengguunakan parameter iterasi. Skenario 1 menggunakan 1.000.000 iterasi,

skenario 2 menggunakan 2.000.000 iterasi dan skenario 3 menggunakan 5.000.000 iterasi. Untuk setiap iterasi dilakukan percobaan sebanyak 11 kali mengacu pada penelitian sebelumnya [6].

6.2.1. Skenario 1 : Iterasi 1.000.000

Pada skenario 1 untuk semester ganjil ini, percobaan dilakukan sebanyak 11 kali dengan iterasi total 1.000.000. Iterasi terbagi menjadi 2 yaitu 500 iterasi pada *global search* dan 500 iterasi pada *local search* untuk setiap *low level heuristic.*



Gambar 6.1 Semester Ganjil 1000000 Iterasi

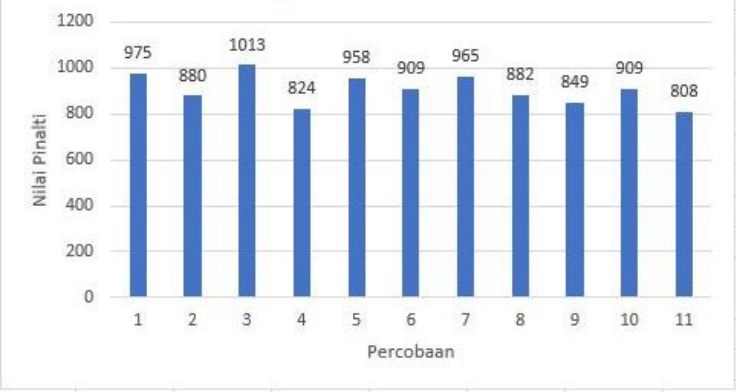
Gambar 6.1 diatas menunjukkan hasil dari 11 percobaan yang masing-masing dilakukan selama 1.000.000 iterasi. Nilai terbesar pinalti adalah 1228, sedangkan yang terkecil adalah 915. Rata-rata dari 11 percobaan tersebut adalah 1023. Sehingga dapat dilihat bahwa terdapat penurunan nilai pinalti jika dibandingkan dengan nilai pinalti manual dan solusi awal seperti pada Tabel 6.2 dibawah ini

Tabel 6.2 Perbandingan Pinalti Ganjil 1000000 Iterasi

Jadwal	Manual	Solusi Awal	Rata-rata Iterasi 1.000.000
Nilai Pinalti	2675	2490	1023

6.2.2. Skenario 2 : Iterasi 2.000.000

Skenario kedua menggunakan iterasi sebayak 2.000.000 untuk melihat perbedaan dari iterasi sebelumnya. Sama seperti sebelumnya, iterasi kali ini juga dibagi menjadi dua yaitu 2500 untuk *global search* dan 500 untuk *local search*.



Gambar 6.2 Semester Ganjil 2000000 Iterasi

Gambar 6.2 diatas menunjukkan hasil dari 11 percobaan yang masing-masing dilakukan selama 2.000.000 iterasi. Nilai terbesar pinalti adalah 1013, sedangkan yang terkecil adalah 808. Rata-rata dari 11 percobaan tersebut adalah 907. Sehingga dapat dilihat bahwa terdapat penurunan nilai pinalti

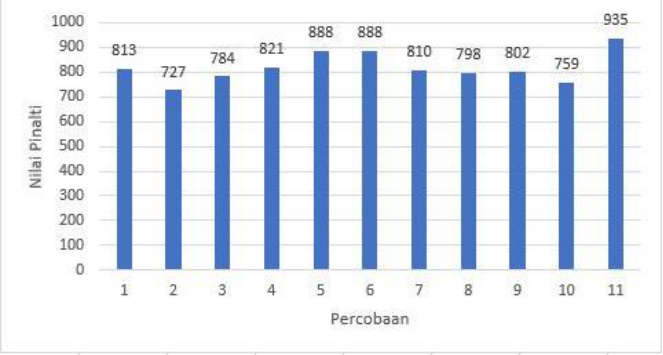
jika dibandingkan dengan nilai pinalti manual dan solusi awal seperti pada Tabel 6.3 dibawah ini

Tabel 6.3 Perbandingan Pinalti Ganjil 2000000 Iterasi

Jadwal	**Manual**	**Solusi Awal**	**Rata-rata Iterasi 2.000.000**
Nilai Pinalti	2675	2490	907

6.2.3. Skenario 3 : Iterasi 5.000.000

Skenario yang terakhir menggunakan 5.000.000 iterasi dengan pembagian 2500 iterasi *global search* dan 500 iterasi *local search* untuk 5 *low level heuristic.*



Gambar 6.3 Semester Ganjil 5000000 Iterasi

Gambar 6.3 diatas menunjukkan hasil dari 11 percobaan yang masing-masing dilakukan selama 5.000.000 iterasi. Nilai terbesar pinalti adalah 935, sedangkan yang terkecil adalah 727. Rata-rata dari 11 percobaan tersebut adalah 820. Sehingga dapat dilihat bahwa terdapat penurunan nilai pinalti

jika dibandingkan dengan nilai pinalti manual dan solusi awal seperti pada Tabel 6.4 dibawah ini

Tabel 6.4 Perbandingan Pinalti Ganjil 5000000 Iterasi

Jadwal	**Manual**	**Solusi Awal**	**Rata-rata Iterasi 5.000.000**
Nilai Pinalti	2675	2490	820

6.3. Hasil Implementasi Penjadwalan Semester Genap

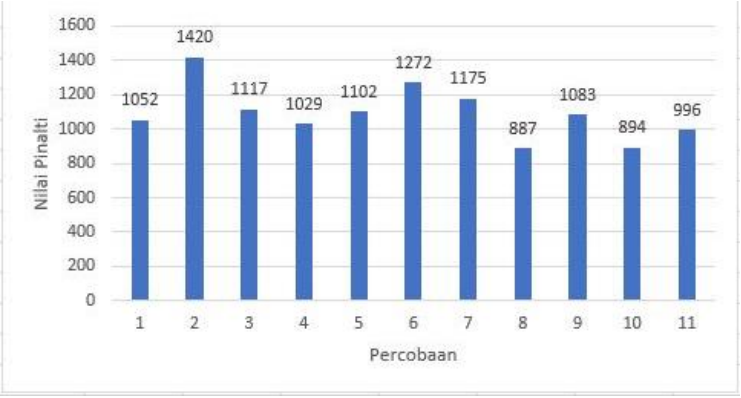
Sama seperti halnya semester ganjil, hasilimplementasi pada semester genap juga akan dibandingkan menggunakan variabel nilai pinalti yang didapat untuk masing masing solusi yang dihasilkan. Nilai pinalti menunjukkan seberapa banyak solusi tersebut dapat memenuhi kriteria *soft constraint* yang ada. Skenario yang dibuat mengguunakan parameter iterasi. Skenario 1 menggunakan 1.000.000 iterasi, skenario 2 menggunakan 2.000.000 iterasi dan skenario 3 menggunakan 5.000.000 iterasi.

6.3.1. Skenario 1 : Iterasi 1.000.000

Pada skenario 1 untuk semester genap ini, percobaan dilakukan sebanyak 11 kali dengan iterasi total 1.000.000. Iterasi terbagi menjadi 2 yaitu 500 iterasi pada *global search* dan 500 iterasi pada *local search* untuk setiap *low level heuristic.*

Gambar 6.4 menunjukkan hasil dari 11 percobaan yang masing-masing dilakukan selama 1.000.000 iterasi. Nilai terbesar pinalti adalah 1420, sedangkan yang terkecil adalah 887. Rata-rata dari 11 percobaan tersebut adalah 1093. Sehingga dapat dilihat bahwa terdapat penurunan nilai pinalti

jika dibandingkan dengan nilai pinalti manual dan solusi awal seperti pada Tabel 6.5.



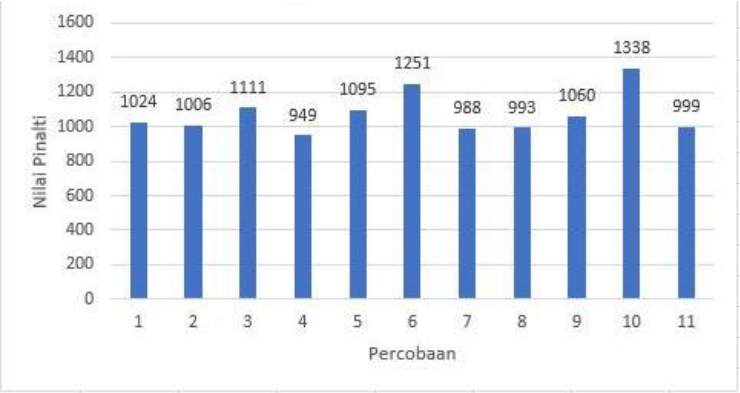
Gambar 6.4 Semester Genap 1.000.000 Iterasi

Tabel 6.5 Perbandingan Pinalti Genap 1000000 Iterasi

Jadwal	**Manual**	**Solusi Awal**	**Rata-rata Iterasi 1.000.000**
Nilai Pinalti	1984	2311	1093

6.3.2. Skenario 2 : Iterasi 2.000.000

Skenario kedua menggunakan iterasi sebayak 2.000.000 untuk melihat perbedaan dari iterasi sebelumnya. Sama seperti sebelumnya, iterasi kali ini juga dibagi menjadi dua yaitu 2500 untuk *global search* dan 500 untuk *local search*.



Gambar 6.5 Semester Genap 2.000.000 Iterasi

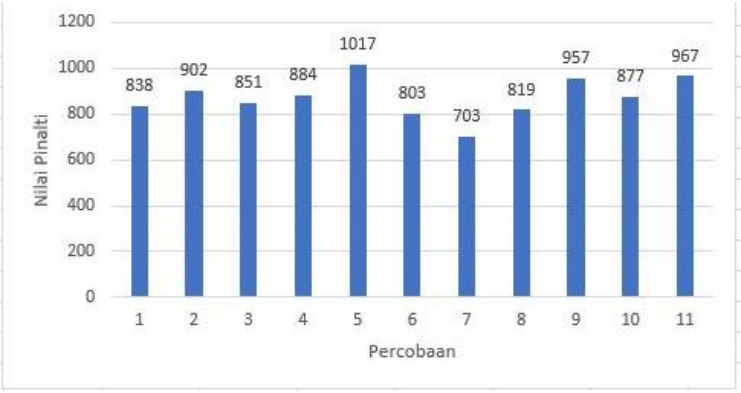
Gambar 6.5 diatas menunjukkan hasil dari 11 percobaan yang masing-masing dilakukan selama 2.000.000 iterasi. Nilai terbesar pinalti adalah 1338, sedangkan yang terkecil adalah 949. Rata-rata dari 11 percobaan tersebut adalah 1074. Sehingga dapat dilihat bahwa terdapat penurunan nilai pinalti jika dibandingkan dengan nilai pinalti manual dan solusi awal seperti pada Tabel 6.6 dibawah ini

Tabel 6.6 Perbandingan Pinalti Genap 2000000 Iterasi

Jadwal	**Manual**	**Solusi Awal**	**Rata-rata Iterasi 2.000.000**
Nilai Pinalti	1984	2311	1074

6.3.3. Skenario 3 : Iterasi 5.000.000

Skenario yang terakhir menggunakan 5.000.000 iterasi dengan pembagian 2500 iterasi *global search* dan 500 iterasi *local search* untuk 5 *low level heuristic.*



Gambar 6.6 Semester Genap 5.000.000 Iterasi

Gambar 6.6 diatas menunjukkan hasil dari 11 percobaan yang masing-masing dilakukan selama 5.000.000 iterasi. Nilai terbesar pinalti adalah 1017, sedangkan yang terkecil adalah703. Rata-rata dari 11 percobaan tersebut adalah 874. Sehingga dapat dilihat bahwa terdapat penurunan nilai pinalti jika dibandingkan dengan nilai pinalti manual dan solusi awal seperti pada Tabel 6.7 dibawah ini

Tabel 6.7 Perbandingan Pinalti Genap 5000000 Iterasi

Jadwal	**Manual**	**Solusi Awal**	**Rata-rata Iterasi 5.000.000**
Nilai Pinalti	1984	2311	874

6.4. Perbandingan Hasil Uji Coba

Setelah melakukan beberapa percobaan dengan melakukan perubahan parameter, maka perlu dilakukan perbandingan hasil untuk dapat mengetahui perubahan yang terjadi pada setiap percobaan. Hal pertama yang akan dibandingan adalah

dari segi jumlah iterasi yang dilakukan untuk 11 percobaan. Yang kedua merupakan perbandingan menggunakan algoritma lain dengan menggunakan 1.000.000 iterasi.

6.4.1. Perbandingan Hasil Uji Coba Iterasi

Perbandingan hal yang pertama adalah terkait iterasi yang dilakukan untuk masing-masing jadwal yaitu ganjil dan genap. Hal ini akan membantu untuk mengetahui pengaruh jumlah iterasi terhadap nilai pinalti yang dihasilkan. Perbandingan dilakukan untuk 11 percobaan dengan melihat nilai terbesar, nilai terkecil dan rata-rata untuk setiap iterasinya.

Tabel 6.8 Perbandingan Iterasi Semester Ganjil

Percobaan	Hasil Pinalti TS Ganjil		
	1000000	2000000	5000000
1	1025	975	813
2	996	880	727
3	1120	1013	784
4	1166	824	821
5	967	958	888
6	924	909	888
7	980	965	810
8	1228	882	798
9	986	849	802
10	915	909	759
11	946	808	935
MAX	1228	1013	935
MIN	915	808	727
AVERAGE	1023	906.5455	820

Tabel 6.8 menunjukkan hasil perbandingan iterasi dari jadwal semester ganjil. Data tersebut menunjukkan bahwa nilai rata-rata pinalti terkecil adalah pada iterasi 5.000.000 yaitu sebesar 820. Nilai pinalti terkecil ada pada percobaan ke 2 iterasi 5.000.000.

Tabel 6.9 Perbandingan Iterasi Semester Genap

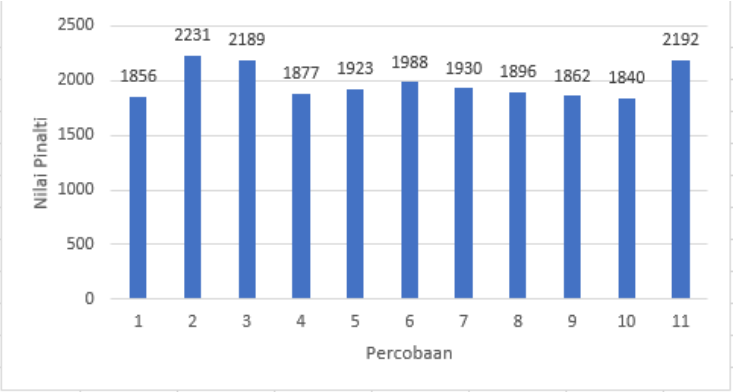
Percobaan	Hasil Pinalti TS		
	1000000	2000000	5000000
1	1052	1024	838
2	1420	1006	902
3	1117	1111	851
4	1029	949	884
5	1102	1095	1017
6	1272	1251	803
7	1175	988	703
8	887	993	819
9	1083	1060	957
10	894	1338	877
11	996	999	967
MAX	1420	1338	1017
MIN	887	949	703
AVERAGE	1093.3636	1074	874

Tabel 6.9 menunjukkan hasil perbandingan iterasi dari jadwal semester genap. Data tersebut menunjukkan bahwa nilai rata-rata pinalti terkecil adalah pada iterasi 5.000.000 yaitu sebesar 874. Nilai pinalti terkecil ada pada percobaan ke 7 iterasi 5.000.000. Berdasarkan ciri-ciri yang muncul pada data

semester ganjil dan genap, maka dapat ditarik kesimpulan bahwa semakin banyak iterasi yang dilakukan maka semakin bagus pula nilai pinalti yang dihasilkan.

6.4.2. Perbandingan Algoritma *Hill Climbing*

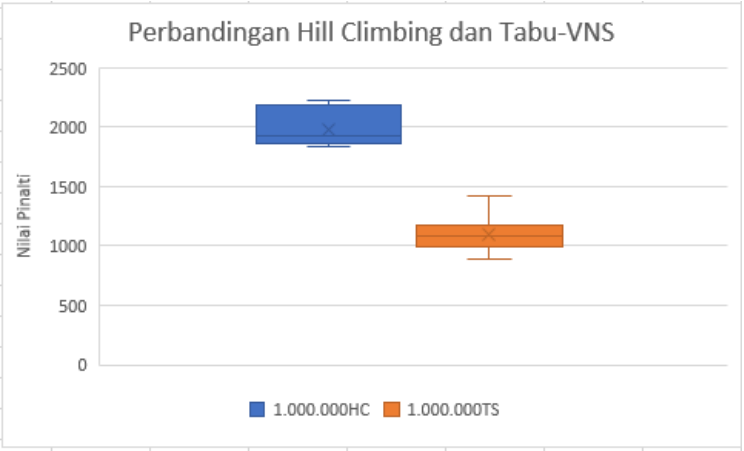
Perbandingan dengan algoritma lain digunakan untuk melihat performa dari algoritma *tabu-variable neighborhood search*. Pembanding tersebut dipilih karena memiliki beberapa kesamaan dengan algoritma yang digunakan untuk implementasi, namun algoritma *tabu-variable neighborhood search* memiliki kompleksitas yang lebih. Uji coba dengan algoritma *hill climbing* dilakukan pada jadwal semester genap sebanyak 11 percobaan dengan 1.000.000 iterasi untuk setiap percobaannya.



Gambar 6.7 Hill Climbing Semester Genap

Gambar 6.7 menunjukkan bahwa nilai pinalti terbesar adalah 2231 yang terletak pada percobaan 2, sedangkan nilai terkecil ada pada percobaan 10 yaitu sebesar 1840. Hasil percobaan diatas memiliki nilai rata-rata sebesar 1980. Untuk dapat

melihat perbandingan persebaran nilai dari masing-masing algoritma, maka dilakukan visualisasi dengan menggunakan *box plot* seperti Gambar 6.8 dibawah ini. Berdasarkan grafik tersebut dapat dilihat persebaran yang dihasilkan algoritma Tabu-VNS lebih baik dikarenakan memiliki persebaran nilai pinalti yang lebih kecil jika dibandingkan dengan persebaran nilai pinalti milik algoritma *Hill Climbing.*



Gambar 6.8 Perbandingan Hill Climbing dan Tabu-VNS

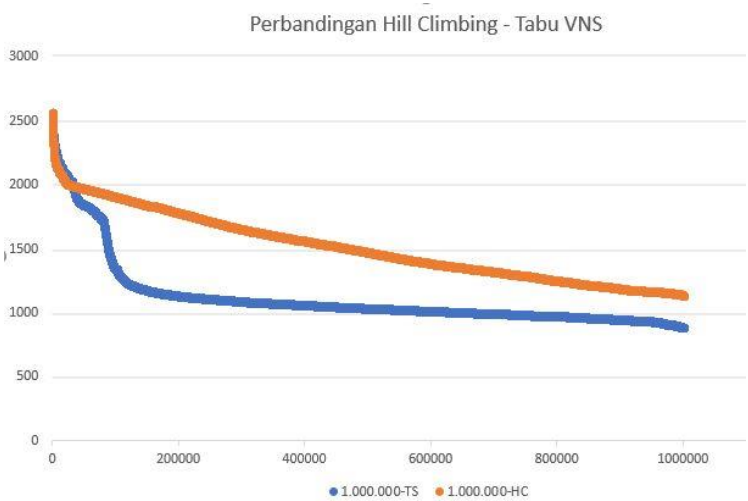
Tabel 6.10 Detail perbandingan AlgoritmaMenunjukan detail nilai pinalti selama 11 kali percobaan, disertai dengan nilai maksimal, minimal dan juga rata-rata.

Tabel 6.10 Detail perbandingan Algoritma

Percobaan	1000000 HC	1000000 Tabu-VNS
1	1856	1052
2	2231	1420
3	2189	1117

Percobaan	1000000 HC	1000000 Tabu-VNS
4	1877	1029
5	1923	1102
6	1988	1272
7	1930	1175
8	1896	887
9	1862	1083
10	1840	894
11	2192	996
MAX	2231	1420
MIN	1840	887
AVERAGE	1980.364	1093.364

Gambar 6.9 menunjukkan perbandingan detail selama 1.000.000 iterasi antara algoritma *Hill Climbing* dan *Tabu-VNS*. Berdasarkan gambar tersebut dapat dilihat bahawa kedua algoritma tersebut mengalami penurunan selama iterasi, namun penurunan algoritma *Hill Climbing* tidak seoptimal algoritma *Tabu-VNS.*



Gambar 6.9 Hasil Perbandingan Iterasi *Hill Climbing* dan *Tabu-VNS*

BAB VII
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan rangkuman singkat yang dapat disimpulkan dari penelitian ini. Terdapat saran dari penulis yang nantinya diharapkan dapat membantu dalam meningkatkan hasil pada penelitian selanjutnya.

7.1. Kesimpulan

Kesimpulan yang dapat diambil dari penelitian ini antara lain adalah :

- Penerapan Algoritma *Tabu-Variable Neighborhood Search based hyper heuristic* dapat digunakan untuk membuat jadwal mata kuliah Departemen Sistem Informasi yang optimal dan memenuhi *hard constraint* yang ada.
- Penerapan Algoritma *Tabu-Variable Neighborhood Search based hyper heuristic* dapat membuat jadwal mata kuliah lebih baik karena mampu menurunkan nilai pinalti pada semester ganjil sebesar 1948 pinalti dan semester genap sebesar 1281 pinalti untuk hasil optimasi. Hal tersebut menunjukkan bahwa jadwal baru yang dihasilkan telah mengurangi pelanggaran terhadap *soft cosntraint* yang ada

7.2. Saran

Berdasarkan hasil dan kesimpulan diatas, saran yang dapat diberikan untuk penelitian selanjutnya adalah sebagai berikut:

- Penelitian ini hanya menggunakan metode *tabu search hyper heuristics* dimana algoritma *tabu search* masih

memiliki banyak kelemahan diantaranya proses iteriasi yang dibutuhkan sangat banyak untuk mencari solusi. Untuk mencari nilai yang lebih optimal dengan cara yang lebih efisien maka penelitihan selanjutnya dapat mencoba metode atau algoritma yang lebih *advance* dari *tabu search*.

- Pada penelitian ini hanya menggunakan low level heuristics sebanyak dua yaitu move dan swap. Untuk menemukan variansi solusi yang optimal maka perlu adanya *low level heuristics* lebih agar proses optimasi dapat memilih lebih banyak heuristics sehingga hasil optimasi lebih dapat dioptimalkan.
- Penelitian ini dilakukan hanya mempertimbangkan dari sisi mahasiswa, untuk selajutnya dapat ditambahkan pertimbangan dari dosen pula seperti kebisaan dosen mengajar pada waktu tertentu.

DAFTAR PUSTAKA

[1] S. Abdullah, "Heuristic Approaches For University Timetabling Problems," The University of Nottingham for the degree of Doctor of Philosophy, 2006.

[2] C. H. Aladag, G. Hocaoglu, and M. A. Basaran, "The effect of neighborhood structures on tabu search algorithm in solving course timetabling problem," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 12349–12356, 2009.

[3] O. Rossi -Doria *et al.*, "A comparison of the performance of different metaheuristics on the timetabling problem," no. IDSIA-18-02, pp. 329–351, 2002.

[4] M. A. Muslim and S. H. Pramono, "Implementasi Algoritma Genetik-Tabu Search dalam Optimasi Penjadwalan Perkuliahan," *J. EECCIS*, vol. 10, no. 2, pp. 45–50, 2016.

[5] P. C. Bwananesia, *Optimasi Penjadwalan Ujian Otomatis dengan Menggunakan Algoritma Greedy-Late Acceptance-Hyper Heuristic*. 2018.

[6] Dian Kusumawardani, *OPTIMASI PENJADWALAN MENGGUNAKAN ALGORITMA GREEDY SIMULATED ANNEALING HYPER HEURISTIC*. 2018.

[7] G. Y. Hutama, *Optimasi Penjadwalan Ujian Otomatis Dengan Menggunakan Algoritma Greedy Hill Climbing Hyper-Heuristic*. 2018.

[8] W. A. Puspaningrum, A. Djunaidy, and R. A. Vinarti, "Penjadwalan Mata Kuliah Menggunakan Algoritma

Genetika di Jurusan Sistem Informasi ITS," *J. Tek. Pomits*, vol. 2, no. 1, pp. 127–131, 2013.

[9] A. (Universitas T. Rochman, "Penjadwalan Kuliah Menggunakan Metode Constraints Programming Dan Simulated Annealing," *Snati*, vol. 2012, no. Snati, pp. 15–16, 2012.

[10] E. L. Lawler, "Combinatorial Optimization : Networks and Matroids," *Comb. Optim. networks matroids*, pp. 1–374, 1976.

[11] A. Schaerf, "Survey of automated timetabling," *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 87–127, 1999.

[12] E. Verhees, *A Heuristic Approach To University Course Timetabling*. 2013.

[13] M. Kalender, A. Kheiri, E. Özcan, and E. K. Burke, "A greedy gradient-simulated annealing selection hyper-heuristic," *Soft Comput.*, vol. 17, no. 12, pp. 2279–2292, 2013.

[14] B. and M. Burke, Edmund and MacCloumn and R. Amnon and Petrovic, Sanja and Qu, "A Graph-Based Hyper-Heuristic for Educational," *Eur. J. Oper. Res.*, vol. 176, pp. 177–192, 2007.

[15] J. A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E. K. Burke, "Effective learning hyper-heuristics for the course timetabling problem," *Eur. J. Oper. Res.*, vol. 238, no. 1, pp. 77–86, 2014.

[16] A. Muklason, "Solver Penjadwal Ujian Otomatis Dengan Algoritma Maximal Clique dan Hyper-heuristics," pp. 18–19, 2017.

[17] K. Setemen, "Optimasi Generate Jadwal Mata Kuliah Menggunakan Algoritma Genetika dan Tabu Search,"

in *Seminar Internasional Revitalisasi Pendidikan Kejuruan dalam Pengembangan SDM Nasional*, pp. 783–791.

[18] D. Pham and D. Karaboga, *Intelligent Optimisation Techniques – Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. 2000.

[19] S. Abdullah, E. K. Burke, and B. Mccollum, "An investigation of variable neighbourhood search for university course timetabling," *Proc. 2nd Multi-disciplinary Interna- tional Conf. Sched. Theory Appl.*, pp. 413–427, 2005.

[20] P. Hansen and N. Mladenovic, "Variable Neighborhood Decomposition Search," *J. Heuristics*, vol. 7, pp. 335–350, 2001.

[21] S. Stepanenko, "Global Optimization Methods based on Tabu Search," *Diss. Thesis*, vol. Univeristy, 2008.

[22] K. Socha, J. Knowles, and M. Sampels, "A MAX - MIN Ant System for the University Course Timetabling Problem," *Ant algorithms*, pp. 1–13, 2002.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Penulis Tugas Akhir ini bernama Redian Galih Irianti. Lahir di Tuban, 4 Agustus 1996. Anak kedua dari tiga bersaudara pasangan Dwi Irianto dan Sudarwati. Penulis menyelesaikan pendidikan Sekolah Dasar di SDN Kebonsari 1 Tuban pada tahun 2009. Pada tahun itu juga penulis melanjutkan Pendidikan di SMP Negeri 1 Tuban dan tamat pada tahun 2011 kemudian melanjutkan Sekolah Menengah Atas di SMA Negeri 1 Tuban pada tahun 2011 dan selesai pada tahun 2014. Pada tahun 2014 penulis melanjutkan pendidikan di perguruan tinggi negeri, tepatnya di Institut Teknologi Sepuluh Nopember Fakultas Teknologi Informasi dan Komunikasi pada Program Studi Sistem Informasi. Sekarang ini penulis tengah berada pada semester delapan perkuliahan.Selama perkuliahan, penulis aktif sebagai panitia kegiatan baik tingkat jurusan, fakultas maupun Institut dengan menjadi panitia Information System Expo (ISE), FTIf Journey, *Basic Media Schooling*2017 dan GERIGI ITS 2015-2016. Penulis juga aktif berorganisasi di Himpunan Mahasiswa Sistem Informasi.

Di Departemen Sistem Informasi, penulis tertarik untuk mengambil bidang minat Rekayasa Data dan Inteligensia Bisnis. Untuk mengetahui informasi lebih lanjut terkait penulis maupun penelitian ini, dapat menghubungi melalui email rediangalih96@gmail.com.

LAMPIRAN A: Hasil Wawancara

Interview Protocol	
Sekretaris Departemen Sistem Informasi ITS	
Informasi Interview	
Interviewer	Redian Galih Irianti
Narasumber	Feby Artwodini Muqtadiroh, S.Kom, M.T.
Hari tanggal	Kamis, 1Maret2018
Pukul	08.30-09.30
Lokasi	Departemen Sistem Informasi
Informasi Narasumber	
Nama	Feby Artwodini Muqtadiroh, S.Kom, M.T.
Jabatan	Sekretaris Program Studi Sarjana (S1) Sistem Informasi, Departemen Sistem Informasi, FTIK, ITS
Instansi	Departemen Sistem Informasi
Penjelasan	Interview ini bertujuan untuk menggali proses penjadwalan mata kuliah di

Interview	Program Studi Sarjana (S1) Sistem Informasi pada Semester Genap dan Semester Ganjil, Tahun Ajaran 2017/2018.Hal ini dimaksudkan agar peneliti bisa mendapatkan gambaran mengenai permasalahan penjadwalan mata kuliah pada Departemen Sistem Informasi
	Dengan melakukan interview peneliti juga diharapkan nisa mendapatkan data-data primer yang dibutuhkan seperti jadwal mata kuliah, jadwal peserta, dll.

NO	**Soal**	**Jawaban**
1	Pada proses penyusunan jadwal mata kuliah siapa saja yang terlibat?	untuk pembuatan jadwal biasanya saya menyusun dulu, sesuai dengan request dari dosen senior, lalu nanti dikonsultasikan ke Pak Soni, kemudian di acc
2	Ada berapakan Tipe staff staff rumah sakit dipekerjakan? (Skil Type)	Dalam penyusunan jadwal mata kuliah, adakah tools yang digunakan untuk mempermudah penyusunannya?
3	erapa lama waktu yang dibutuhkan dalam penyusunan jadwal mata kuliah?	untuk penyusunan jadwal mulai dibicarakan saat rapat akhir tahun, lalu dilakukan pembuatan

		jadwal biasanya memakan waktu 1-2 minggu
4	Bagaimana prosedur (step by step) dalam pembuatan jadwal mata kuliah selama ini pada program studi S1 sistem informasi?	1. Pak Soni membuat kuisioner terkait kelas matpil yang banyak diminati mahasiswa untuk menjadi pertimbangan membuka kelas 2. pada saat rapat pembubaran, memaparkan mata kuliah yang akan dibuka, beserta pembagian dosen pengampu 3. dosen yang memiliki jabatan selain di SI (Seperti Bu Erma wakil dekan, Pak Arif wakil rektor, Dosen Senat, dll) biasanya akan request jadwal yang beliau-beliau bisa mengajar 4. Bu Febby akan menyusun jadwal dengan terlebih dahulu memprioritaskan request dari Dosen berkepentingan

		5. Selanjutnya Bu Febby akan melihat jadwal upmb untuk kemudian dimasukkan dulu. Tidak meletakkan matkul jurusan mahasiswa yang juga mengambil matkul upmb. 6. Setelah itu menyusun jadwal yang memiliki 4 sks supaya tidak berdekatan 7. Lalu menempatkan mata kuliah sisanya pada timeslot yang kosong 8. konsultasi ke Pak Soni 9. Release ke dosen, biasanya ada request yang datang dari dosen terkait jadwal dan sebisa mungkin dipenuhi.
5	Apa sajakah yang diperhatikan atau dipertimbangkan dalam penyusunan jadwal mata kuliah?	• Jumlah dosen • Jumlah dan kapasitas ruangan yang digunakan (berapa ruangan? Apa saja?) iyaa, untuk setiap

		semester terdiri dari 98 kelas/semester • Mata kuliah (ada berapa yang dibuka dalam satu semester itu?) 30 sekian, kelas yang dibuka 4 per matkul wajib • Request dosen yang tiba-tiba • Jadwal upmb
6	Apa yang menjadi pertimbangan untuk membuka suatu kelas untuk mata kuliah tertentu? Dosen pengampu?	• Dilihat dari mata kuliah yang sering mengulang • dari kuisioner yang disebar Pak Soni • untuk jumlah kelas yang akan dibuka juga mempertimbangkas ketersediaan dosen pengampu
7	Bagaimana memastikan bahwa jadwal kuliah tiap angkatan tidak mengalami bentrok? (validasi)	Mengecek manual
8	Bagaimana jadwal kuliah dapat mengakomodasi permintaan khusus dari	• Untuk request dari dosen yang memiliki urusan di luar SI,

	dosen? Contohnya dosen x hanya bisa mengajar pagi atau siang?	didahulukan dulu • Untuk request dari dosen SI sebisa mungkin juga dipenuhi.
9	Adakah perubahan/pengembangan yang dilakukan tiap semesternya dalam penyusunan jadwal mata kuliah? Jika ada, apa saja perubahan yang dilakukan?	perubahannya dari segi dosen pengajar. Jika dulu di jadwal, untuk nama dosen masih sering ada tanda "/" maka untuk sekarang mengurangi hal tersebut, karena ternyata sebenarnya dosen tidak terlalu suka yang seperti itu.
10	Jika terdapat ketidaksesuaian jadwal, apakah dapat dilakukan pembenahan secara cepat dari jadwal lama?	untuk pembenahan diusahakan bisa diselesaikan dalam waktu sehari itu saja. Namun untuk permasalahan kelas bentrok oleh mahasiswa karena berpindah kelas, hal itu biasanya mahasiswa meminta langsung untuk pindah kelas yang lain, tanpa harus mengubah jadwal asli.
11	Adakah aturan dalam menentukan jadwal mata	• Mahasiswa tidak dapat berkuliah 2 atau

	kuliah? Jika ada, apa sajakah aturan tersebut? (seperti pemilihan jadwal mata kuliah disesuaikan angkatan)	lebih mata kuliah diwaktu yang sama • Dosen tidak dapat mengajar 2 atau lebih mata kuliah diwaktu yang sama • Satu kelas hanya boleh ada satu mata kuliah didalamnya pada kurun waktu tertentu • Jumlah siswa harus sama dengan atau kurang dari kapasitas kelas • Jadwal jurusan menyesuaikan jadwal upmb • Request dari dosen berkepentingan harus diprioritaskan • Pembagian pengajar didasari pada suatu patokan atau ukuran • Lab digunakan untuk matakuliah tertentu • Mahasiswa maksimal berkuliah 2 matkul tiap harinya • Request dosen biasa sebisa mungkin

		diakomodasi
12	Apa sajakah kesulitan yang dialami saat melakukan penyusunan jadwal mata kuliah? Dari dulu hingga sekarang !	• adanya request dari dosen biasa • adanya kelas yang mengulang
13	Seberapa berpengaruh kesuksesan penyusunan jadwal mata kuliah bagi departemen sistem informasi?	untuk mahasiswa, jadwal yang ideal (maksimal kuliah hanya 2/ hari) akan membantu mahasiswa untuk tetap konsentrasi saat belajar
14	Apakah pernah ada usulan sebelumnya terkait otomasi penyusunan jadwal mata kuliah? Bagaimana kesannya?	dulu sudah pernah dibicarakan terkait optimasi penjadwalan mata kuliah, tapi hingga sekarang belum menemukan tools yang tepat. Karena batasan-batasan yang ada itu berubah tiap semesternya, dan batasan yang tiba-tiba muncul juga membuat sulit untuk di optimasi. Lebih ke pembuatan jadwal melibatkan sisi kemanusiaan.
15	Apa harapan kedepan	Semoga kalo memang

	untuk penyusunan jadwal mata kuliah pada departemen sistem informasi? (dapat dilakukan secara otomatis)	bisa dioptimasi, juga tetap bisa memenuhi segala
16	Pernahkan melibatkan pendapat mahasiswa dalam melakukan penyusunan jadwal mata kuliah?	mahasiswa biasanya dilibatkan dalam hal penyusunan jadwal untuk matpil.

LAMPIRAN B : Kebutuhan Data

Tabel B- 1 Jadwal Semester Ganjil *full*

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
Senin	07.30-10.00	PBO [C]	MPTI [B]	PSSI [B]	TTI [A]	TOST [A]	SS [A]	MO [B]	PSDP [C]	-	-	MO [A]
		3 sks/ sem 3	4 sks/ sem 5	3 sks/ sem 7	3 sks/ sem 7	3 sks/ sem 7	2 sks/ sem 5	3 sks/ sem 1	4 sks/ sem 5			3 sks/ sem 1
		FJ	AH	KG	MW	HS	ES	ED	IH			AW
	10.15-12.45	EP [A]	MPTI [A]	MPTI [C]	AUDIT [A]	-	RO [D]	KCB [B]	KCB [E]	PBO [A]	-	PSDP [B]
		2 sks/	4 sks/	4 sks/	3 sks/		3 sks/sem	3 sks/sem	3 sks/sem	3 sks/		4 sks/sem

B-2

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
		sem 7	sem 5	sem 5	sem 7		5	7	7	sem 3		5
		NA	AH	TE	KG		ED	RH	FM	FJ		AW
	13.30-16.00	EP [B]	STAT [B]	MPTI [D]	DMPB [A]			KCB [C]	KCB [D]			
		2 sks/sem 7	4 sks/sem 3	4 sks/sem 5	4 sks/sem 3	-	-	3 sks/sem 7	3 sks/sem 7	-	-	-
		NA	WA	FA	MW			RH	FM			
Selasa	07.30-10.00	PBD [A]	RO [B]	PSSI [C]	SC [A]	MO [C]	PKETI [A]	MLTI [C]	DDPL [B]	DDPL [C]		MO [D]
		3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem		3 sks/sem

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
		3	5	7	6	1	7	5	3	3		1
		AD	WA	KG	AT	ED	HM	TD	HS	FA		AW
	10.15-12.45	PSDP [D]	PKETI [B]	PSSI [A]	PBO [D]	EP [Q]	KPPL [B]	FORDIG [A]	BP [A]	BP [B]		DMPB [B]
		4 sks/sem 5	3 sks/sem 7	3 sks/sem 7	3 sks/sem 3	2 sks/sem 7	3 sks/sem 5	3 sks/ sem 7	4 sks/sem 1	4 sks/sem 1	-	4 sks/sem 3
		IH	HM	KG	FS	NA	SH	BC	RK	FJ/NF		AS
	13.30-16.00	PBD [C]	BP [C]	MLTI [D]	PBO [B]	KAI [A]	KPPL [C]	MATDIS [A]	BP [D]	-	TEKPER [A]	PSDP [A]
		3 sks/sem	4 sks/sem	3 sks/sem	3 sks/sem	4 sks/sem	3 sks/sem	3 sks/sem	4 sks/sem		3 sks/ sem 7	4 sks/sem

B-4

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
		3	1	5	3	4	5	1	1			5
		RP	RK	TD	FJ	BC	SH	TE	NF		WA	AW
Rabu	07.30-10.00	PKETI [C]	RO [A]	MPTI [B]	RO [C]	-	KCB [A]	DDPL [A]	-	MATDIS [C]	-	STI [B]
		3 sks/sem 7	3 sks/sem 5	4 sks/sem 5	3 sks/sem 5		3 sks/sem 7	3 sks/sem 3		3 sks/sem 1		3 sks/sem 1
		HM	WA	AH	ED		RH	HS		AM		AS
	10.15-12.45	PBD [B]	PKB [A]	MPTI [A]	STI [C]	-	STAT [A]	DMPB [D]	PSSI [D]	MRTI [A]	-	PSDP [B]
		3 sks/sem	3 sks/sem	4 sks/sem	3 sks/sem		4 sks/sem	4 sks/sem	3 sks/sem	3 sks/sem		4 sks/sem

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
		3	7	5	1		3	3	7	6		5
		RP	AO	AH	NF/IH		RK	AP	AN	TE/HM		AW
	13.30-16.00	PBD [D]	STAT [B]	MPTI [D]	E-BIS [A]	ASD [A]	PSDP [C]	DMPB [C]	PSSI [E]	-	-	STI [A]
		3 sks/sem 3	4 sks/sem 3	4 sks/sem 5	3 sks/sem 7	3 sks/sem 2	4 sks/sem 5	4 sks/sem 3	3 sks/sem 7			3 sks/sem 1
		RH	WA	FA	MW	RP	IH	AP	AN			AS
Kamis	07.30-10.00	PDAB [A]	KPPL [D]	MPTI [C]	SS [B]	STI [D]	BP [A]	UPMB	UPMB	BP [B]		PSDP [A]
		3 sks/sem	3 sks/sem	4 sks/sem	2 sks/sem	3 sks/sem	4 sks/sem			4 sks/sem		4 sks/sem

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
		7	5	5	5	1	1			1		5
		ED	HS	TE	AP	FA	RK			FJ/NF		AW
	10.15-12.45	DMPB [A]	KPPL [A]	PSDP [D]	SS [C]	KAI [A]	DMJK [D]	UPMB	UPMB	-	-	DMPB [B]
		4 sks/sem 3	3 sks/sem 5	4 sks/sem 5	2 sks/sem 5	4 sks/sem 4	3 sks/sem 3					4 sks/sem 3
		MW	SH	IH	AP	BC	NF					AS
	13.30-16.00	MPITI [A]	DMPB [D]	STAT [A]	MLTI [B]		DMJK [B]	SPK [B]	MATDIS [B]	MATDIS [D]	-	-
		3 sks/sem	4 sks/sem	4 sks/sem	3 sks/sem		3 sks/sem	3 sks/sem	3 sks/sem	3 sks/sem		

Hari	Jam	TC101	TC103	TC104	TC105	TC105 A	TC106	TC107	TC108	Aula	Studio Lt.1	Studio Lt.2
		6	3	3	5		3	7	1	1		
		SH	AP	RK	AN		BC	IH	TE	AM		
Jumat	13.00-15.30	SPK [A]	DMPB [C]	BP [C]	BP [D]		MLTI [A]	DMJK [C]	DMJK [A]	-	-	-
		3 sks/sem 7	4 sks/sem 3	4 sks/sem 1	4 sks/sem 1		3 sks/sem 5	3 sks/sem 3	3 sks/sem 3			
		FM	AP	RK	NF		AN	NA	BC			

Tabel B- 2 Jadwal semester genap *full*

HARI	JAM	TC101	TC103	TC104	TC105	TC105A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2

HARI	JAM	TC101	TC103	TC104	TC105	TC105A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2
SENIN	07.30 - 10.00	MRPHP [D]	SC [B]	DBD [B]	PSO [C]		ARSITI [D]	DMPB [A]	TKTI [A]			
		SEM 6,2015	SEM 6,2015	SEM 4,2016	SEM 2,2017		SEM 2,2017	SEM 3,2016	SEM 6,2015			
		ES	ED	IH	NF		KG	AW	AH			
	10.15 - 12.45	ADPL [A]	ADPL [C]	DBD [D]	TKTI [C]		SC [A]	**AUDIT SI**	TKTI [B]			
		SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 6,2015		SEM 6,2015	SEM 7,2014	SEM 6,2015			
		FA	AS	IH	AN		ED	KG	AH			
	13.30 - 16.00	ADPL [B]	ADPL [D]	SC [C]	TKTI [D]		KPPL [A]					
		SEM	SEM	SEM	SEM		SEM					

HARI	JAM	TC101	TC103	TC104	TC105	TC105A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2
		4,2016	4,2016	6,2015	6,2015		7,2014					
		FA	AS	RK	AN		SH					
SELASA	07.30 - 10.00	PDAB [A]	MRPHP [B]	TTI [D]	PBW [A]	BAPRO [A]	STAT [A]	MRTI [A]	DBD [C]	KAI [B]		ARSITI [B]
		SEM 8,2014	SEM 6,2015	SEM 6,2015	SEM 4,2016	SEM 1,2017	SEM 3,2016	SEM 6,2015	SEM 4,2016	SEM 4,2016		SEM 2,2017
		AD	MW	TD	FJ	NF	WA	HM	IH	BC		KG
	10.15 - 12.45	PBW [B]	PBW [C]	MABD [C]	PBW [D]	ASD [C]	STAT [B]	MKTI [A]	MPITI [A]	KAI [A]	PSO [D]	ARSITI [A]
		SEM 4,2016	SEM 4,2016	SEM 6,2015	SEM 4,2016	SEM 2,2017	SEM 3,2016	SEM 8,2014	SEM 6,2015	SEM 4,2016	SEM 2,2017	SEM 2,2017
		HS	FS	RP	FJ	AM	WA	HM	AO	BC	NF	KG
	0 - 1	KKI [B]	PPB [A]	MABD	KKI	DBD	ASD [A]	MRTI	MABD	KKI [D]		

B-10

HARI	JAM	TC101	TC103	TC104	TC105	TC105A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2
				[B]	[C]	[A]		[C]	[D]			
		SEM 2,2017	SEM 8,2014	SEM 6,2015	SEM 2,2017	SEM 4,2016	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 2,2017		
		AS	HS	RP	FA	RH	RK	TE	FM	AN		
RABU	07.30 - 10.00	ADPL [C]	KAI [D]	MLTI [A]	ARSITI [C]	PKETI [A]	DBD [B]	KKI [A]	IAK [A]	SC [D]	PINT [A]	ADPL [A]
		SEM 4,2016	SEM 4,2016	SEM 5,2015	SEM 2,2017	SEM 7,2014	SEM 4,2016	SEM 2,2017	SEM 8,2014	SEM 6,2015	SEM 8, 2014	SEM 4,2016
		AS	NF	TD	**ED**	AN	IH	AP	RH	AT	NA	FA
	10.15 - 12.45	ADPL [D]	KAI [C]		ADPL [B]	TOST [A]	TTI [A]	MRPHP [C]	MRTI [B]			
		SEM 4,2016	SEM 4,2016		SEM 4,2016	SEM 8,2014	SEM 6,2015	SEM 6,2015	SEM 6,2015			

HARI	JAM	TC101	TC103	TC104	TC105	TC105A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2
		AS	BC		FA	HS	SH	AW	TE			
	13.30 - 16.00	**PBO**	DBD [A]	MRPHP [A]		TTI [B]	TTI [C]		MRTI [D]		TEKPER [A]	
		SEM 3,2016	SEM 4,2016	SEM 6,2015		SEM 6,2015	SEM 6,2015		SEM 6,2015		SEM 8,2014	
		RP	RH	MW		RK	TD		TE		WA	
KAMIS	07.30 - 10.00	**SPK [B]**		DBD[D]	MABD [A]	EP [A]	IMK [C]	STAT [B]	IMK [A]		KAI [B]	DMPB [A]
		SEM 8,2014		SEM 4,2016	SEM 6,2015	SEM 7,2014	SEM 4,2016	SEM 3,2016	SEM 4,2016		SEM 4,2016	SEM 3,2016
		ED		IH	FM	NA	RH	WA	FJ		BC	AW
	5 - 12.4	MATDIS [A]	DBD [C]	MPITI [B]	SPK [A]	EP [Q]	IMK [D]	STAT [A]	IMK [B]	ROL [A]	PSO [B]	ASD [D]

HARI	JAM	TC101	TC103	TC104	TC105	TC105A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2
		SEM 1,2017	SEM 4,2016	SEM 6,2015	SEM 8,2014	SEM 7,2014	SEM 4,2016	SEM 3,2016	SEM 4,2016	SEM 8,2014	SEM 2,2017	SEM 2,2017
		TE	IH	SH	FM	NA	RH	WA	FJ	AM	AP	AW
	13.30 - 16.00	PSO [A]	ASD [B]	MPITI [C]	**EBIS**	BAPRO [A]						KAI [A]
		SEM 2,2017	SEM 2,2017	SEM 6,2015	SEM 8,2014	SEM 1,2017						SEM 4,2016
		AP	RK	SH	AW	NF						BC
JUMAT	07.00 -	**RAPAT DAN KEGIATAN MAHASISWA**										
	13.00 - 15.30		KAI [D]	**KKI [C]**	**KKI [D]**			**KKI [A]**	**KKI [B]**			KAI [C]
			SEM 4,2016	SEM 2,2017	SEM 2,2017			SEM 2,2017	SEM 2,2017			SEM 4,2016

HARI	J AM	TC101	TC103	TC104	TC10 5	TC10 5A	TC106	TC107	TC108	AULA	Studio Lt 1	Studio Lt 2
			NF	FA	AN			AP	AS			BC

Tabel B- 3 Konversi Kode *Event*semester ganjil *full*

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
1	ASD A	001	40	KPPL C	040	79	PSDP A3	079
2	AUDIT A	002	41	KPPL D	041	80	PSDP A4	080
3	BAPRO A1	003	42	MATDIS A	042	81	PSDP B1	081
4	BAPRO A2	004	43	MATDIS B	043	82	PSDP B2	082
5	BAPRO B1	005	44	MATDIS C	044	83	PSDP B3	083
6	BAPRO B2	006	45	MATDIS D	045	84	PSDP B4	084
7	BAPRO C1	007	46	MENRISK A	046	85	PSDP C1	085
8	BAPRO C2	008	47	MLTI A	047	86	PSDP C2	086
9	BAPRO D1	009	48	MLTI B	048	87	PSDP C3	087

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
10	BAPRO D2	010	49	MLTI C	049	88	PSDP C4	088
11	DDPL A	011	50	MLTI D	050	89	PSDP D1	089
12	DDPL B	012	51	MO A	051	90	PSDP D2	090
13	DDPL C	013	52	MO B	052	91	PSDP D3	091
14	DMJK A	014	53	MO C	053	92	PSDP D4	092
15	DMJK B	015	54	MO D	054	93	PSSI A	093
16	DMJK C	016	55	MPITI A	055	94	PSSI B	094
17	DMJK D	017	56	MPTI A1	056	95	PSSI C	095
18	DMPB A1	018	57	MPTI A2	057	96	PSSI D	096
19	DMPB A2	019	58	MPTI B1	058	97	PSSI E	097
20	DMPB B1	020	59	MPTI B2	059	98	RO A	098
21	DMPB B2	021	60	MPTI C1	060	99	RO B	099
22	DMPB C1	022	61	MPTI C2	061	100	RO C	100
23	DMPB C2	023	62	MPTI D1	062	101	RO D	101
24	DMPB D1	024	63	MPTI D2	063	102	SC A	102

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
25	DMPB D2	025	64	PBD A	064	103	SPK A	103
26	EBIS A	026	65	PBD B	065	104	SPK B	104
27	EP A	027	66	PBD C	066	105	SS A	105
28	EP B	028	67	PBD D	067	106	SS B	106
29	EP Q	029	68	PBO A	068	107	SS C	107
30	FORDIG A	030	69	PBO B	069	108	STATIS A1	108
31	KAI A1	031	70	PBO C	070	109	STATIS A2	109
32	KAI A2	032	71	PBO D	071	110	STATIS B1	110
33	KCB A	033	72	PDAB A	072	111	STATIS B2	111
34	KCB B	034	73	PKB A	073	112	STI A	112
35	KCB C	035	74	PKETI A	074	113	STI B	113
36	KCB D	036	75	PKETI B	075	114	STI C	114
37	KCB E	037	76	PKETI C	076	115	STI D	115
38	KPPL A	038	77	PSDP A1	077	116	TEKPER A	116
39	KPPL B	039	78	PSDP A2	078	117	TOST A	117

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
						118	TTI A	118

Tabel B- 4 Konversi kode *event* semester genap *full*

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
1	ADPL A1	001	37	IMK D	037	73	PBO A	073
2	ADPL A2	002	38	KAI A1	038	74	PBW A	074
3	ADPL B1	003	39	KAI A2	039	75	PBW B	075
4	ADPL B2	004	40	KAI B1	040	76	PBW C	076
5	ADPL C1	005	41	KAI B2	041	77	PBW D	077
6	ADPL C2	006	42	KAI C1	042	78	PDAB A	078
7	ADPL D1	007	43	KAI C2	043	79	PINT A	079
8	ADPL D2	008	44	KAI D1	044	80	PKETI A	080
9	ARSITI A	009	45	KAI D2	045	81	PPB A	081
10	ARSITI B	010	46	KKI A1	046	82	PSO A	082
11	ARSITI C	011	47	KKI A2	047	83	PSO B	083

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
12	ARSITI D	012	48	KKI B1	048	84	PSO C	084
13	ASD A	013	49	KKI B2	049	85	PSO D	085
14	ASD B	014	50	KKI C1	050	86	ROL A	086
15	ASD C	015	51	KKI C2	051	87	SC A	087
16	ASD D	016	52	KKI D1	052	88	SC B	088
17	AUDIT	017	53	KKI D2	053	89	SC C	089
18	BP A1	018	54	KPPL A	054	90	SC D	090
19	BP A2	019	55	MABD A	055	91	SPK A	091
20	DBD A1	020	56	MABD B	056	92	SPK B	092
21	DBD A2	021	57	MABD C	057	93	STAT A1	093
22	DBD B1	022	58	MABD D	058	94	STAT A2	094
23	DBD B2	023	59	MATDIS A	059	95	STAT B1	095
24	DBD C1	024	60	MENRISK A	060	96	STAT B2	096
25	DBD C2	025	61	MENRISK B	061	97	TEKPER A	097

No	Event	Kode Event	No	Event	Kode Event	No	Event	Kode Event
26	DBD D1	026	62	MENRISK C	062	98	TKTI A	098
27	DBD D2	027	63	MENRISK D	063	99	TKTI B	099
28	DMPB A1	028	64	MKTI A	064	100	TKTI C	100
29	DMPB A2	029	65	MLTI A	065	101	TKTI D	101
30	EBIS A	030	66	MPITI A	066	102	TOST A	102
31	EP A	031	67	MPITI B	067	103	TTI A	103
32	EP Q	032	68	MPITI C	068	104	TTI B	104
33	IAK A	033	69	MRPHP A	069	105	TTI C	105
34	IMK A	034	70	MRPHP B	070	106	TTI D	106
35	IMK B	035	71	MRPHP C	071			
36	IMK C	036	72	MRPHP D	072			

LAMPIRAN C : Hasil Generate Jadwal Ganjil

Hari	Jam	Mata Kuliah											
Senin	07.30-10.00	DMJK [B]	MATDIS [A]	MATDIS [B]	MATDIS [C]	MATDIS [D]	PBO [D]	PKETI [A]	PKETI [B]	PKETI [C]	TOST [A]		
		3 sks/ sem 3	3 sks/ sem 1	3 sks/ sem 1	3 sks/ sem 1	3 sks/ sem 1	3 sks/ sem 3	3 sks/ sem 7	3 sks/ sem 7	3 sks/ sem 7	3 sks/ sem 7		
	10.15-12.45	ASD [A]	BP [A]	BP [B]	BP [D]	DMJK [A]	DMPB [C]	EP [B]	KCB [C]	KCB [D]	MPTI [D]	PBD [D]	STAT [B]
		3 sks/ sem 2	4 sks/ sem 1	4 sks/ sem 1	4 sks/ sem 1	3 sks/ sem 3	4 sks/ sem 3	2 sks/ sem 7	3 sks/ sem 7	3 sks/ sem 7	4 sks/ sem 5	3 sks/ sem 3	4 sks/ sem 3
	13.30-16.00	BP [A]	BP [B]	BP [C]	BP [D]	PSSI [B]	PSSI [D]	PSSI [E]	SS [A]	SS [B]	SS [C]	STAT [A]	STAT [B]
		4 sks/sem 1	4 sks/sem 1	4 sks/sem 1	4 sks/sem 1	3 sks/sem 7	3 sks/ sem 7	3 sks/sem 7	2 sks/sem 5	2 sks/sem 5	2 sks/sem 5	4 sks/sem 3	4 sks/sem 3
Selasa	07.30-10.00	BP [C]	E-BIS [A]	KCB [E]	MRTI [A]	MLTI [A]	MLTI [B]	MO [A]	MO [B]	MO [D]	PSDP [C]		
		4 sks/sem 1	3 sks/sem 7	3 sks/sem 7	3 sks/sem 6	3 sks/sem 5	3 sks/sem 5	3 sks/sem 1	3 sks/sem 1	3 sks/sem 1	4 sks/sem 5		
	10.15 - 12.45	KPPL [B]	MO [C]	MPTI [C]	PSDP [A]	RO [C]	SPK [A]	SPK [B]	STI [A]	STI [B]	STI [D]		

Hari	Jam	Mata Kuliah											
		3 sks/sem 5	3 sks/sem 1	4 sks/sem 5	4 sks/sem 5	3 sks/sem 5	3 sks/sem 7	3 sks/sem 7	3 sks/sem 1	3 sks/sem 1	3 sks/sem 1		
	13.30-16.00	MLTI [C]	MLTI [D]	MPTI [B]	PDAB [A]	PSDP [A]	STI [C]						
		3 sks/sem 5	3 sks/sem 5	4 sks/sem 5	3 sks/sem 7	4 sks/sem 5	3 sks/sem 1						
Rabu	07.30-10.00	DMPB [A]	MPTI [B]	MPTI [C]	MPTI [D]	TTI [A]							
		4 sks/sem 3	4 sks/sem 5	4 sks/sem 5	4 sks/sem 5	3 sks/sem 7							
	10.15-12.45	AUDIT [A]	EP [A]	KCB [B]	MPTI [A]	PBO [A]	PBO [B]	PBO [C]	PSDP [B]	PSDP [C]	PSDP [D]		
		3 sks/sem 7	2 sks/sem 7	3 sks/sem 7	4 sks/sem 5	3 sks/sem 3	3 sks/sem 3	3 sks/sem 3	4 sks/sem 5	4 sks/sem 5	4 sks/sem 5		
	13.30-16.00	DMPB [D]	MPITI [A]	PBD [B]	PSDP [B]	PSSI [A]	PSSI [C]	STAT [A]					
		4 sks/sem	3 sks/sem	3 sks/sem	4 sks/sem	3 sks/sem	3 sks/sem	4 sks/sem					

Hari	Jam	Mata Kuliah											
		3	6	3	5	7	7	3					
Kamis	07.30-10.00	DDPL [A]	DDPL [B]	DDPL [C]	KCB [A]	KPPL [C]	KPPL [D]	PSDP [A]	PSDP [B]				
		3 sks/sem 3	3 sks/sem 3	3 sks/sem 3	3 sks/sem 7	3 sks/sem 5	3 sks/sem 5	4 sks/sem 5	4 sks/sem 5				
	10.15-12.45	DMJK [C]	DMJK [D]	DMPB [B]	PKB [A]	PSDP [A]	PSDP [B]	RO [C]	SC [A]				
		3 sks/sem 3	3 sks/sem 3	4 sks/sem 3	3 sks/sem 7	4 sks/sem 5	4 sks/sem 5	3 sks/sem 5	3 sks/sem 6				
	13.30-16.00	DMPB [A]	DMPB [B]	DMPB [C]	DMPB [D]	EP [Q]	KPPL [A]	PSDP [D]	TEKPER [A]				
		4 sks/sem 3	4 sks/sem 3	4 sks/sem 3	4 sks/sem 3	2 sks/sem 7	3 sks/sem 5	4 sks/sem 5	3 sks/ sem 7				
Jumat	07.30-10.00	FORDIG [A]	KAI [A]	PBD [A]	PBD [C]	PSDP [A]	PSDP [C]	PSDP [D]	RO [B]				
		3 sks/ sem 7	4 sks/sem 4	3 sks/sem 3	3 sks/sem 3	4 sks/sem 5	4 sks/sem 5	4 sks/sem 5	3 sks/sem 5				
	13.00-15.	KAI [A]	MPTI	PSDP	PSDP								

Hari	Jam	Mata Kuliah											
			[A]	[C]	[D]								
		4 sks/sem 4	4 sks/sem 5	4 sks/sem 5	4 sks/sem 5								

LAMPIRAN D : Hasil Generate Jadwal Genap

Hari	Jam	Mata Kuliah									
Senin	07.30-10.00	DMPB [A]	MKTI [A]	MPITI [A]	MPITI [B]	MPITI [C]	PINT [A]	PSO [C]	ROL [A]	TOST [A]	
		SEM 3,2016	SEM 8,2014	SEM 6,2015	SEM 6,2015	SEM 6,2015	SEM 8, 2014	SEM 2,2017	SEM 8,2014	SEM 8,2014	
	10.15-12.45	ADPL [C]	ARSITI [D]	ASD [C]	DMPB [A]	IMK [A]	PPB [A]	SC [A]	SC [B]	SC [C]	SC [D]
		SEM 4,2016	SEM 2,2017	SEM 2,2017	SEM 3,2016	SEM 4,2016	SEM 8,2014	SEM 6,2015	SEM 6,2015	SEM 6,2015	SEM 6,2015
	13.30-16.00	ASD [A]	ASD [B]	ASD [D]	DBD [A]	DBD [B]	DBD [D]	KPPL [A]	MRPHP [D]	TTI [C]	
		SEM 2,2017	SEM 2,2017	SEM 2,2017	SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 7,2014	SEM 6,2015	SEM 6,2015	
Selasa	07.30-10.00	IMK [B]	IMK [D]	KKI [A]	MABD [B]	MRTI [C]	PSO [D]	STAT [B]			
		SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 2,2017	SEM 3,2016			
	10.15 - 12.45	ARSITI [B]	BAPRO [A]	MRTI [A]	MLTI [A]	PBW [C]	PDAB [A]	STAT [A]	TTI [D]		

Hari	Jam	Mata Kuliah									
		SEM 2,2017	SEM 1,2017	SEM 6,2015	SEM 5,2015	SEM 4,2016	SEM 8,2014	SEM 3,2016	SEM 6,2015		
	13.30-16.00	BAPRO [A]	DBD [A]	DBD [B]	KKI [B]	MRTI [D]	MRPHP [A]	PBO	TEKPER [A]	TTI [B]	
		SEM 1,2017	SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 3,2016	SEM 8,2014	SEM 6,2015	
Rabu	07.30-10.00	ADPL [A]	DBD [C]	DBD [D]	EP [Q]	KAI [B]	MATDIS [A]	PSO [B]	SPK [A]	STAT [A]	
		SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 7,2014	SEM 4,2016	SEM 1,2017	SEM 2,2017	SEM 8,2014	SEM 3,2016	
	10.15-12.45	ADPL [A]	ADPL [B]	ADPL [C]	ADPL [D]	KKI [A]	STAT [B]	TKTI [A]	TKTI [C]	TKTI [D]	
		SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 3,2016	SEM 6,2015	SEM 6,2015	SEM 6,2015	
	13.30-16.00	ADPL [B]	ADPL [D]	DBD [C]	EP [A]	KAI [B]	KAI [C]	KKI [D]	MABD [A]	SPK [B]	
		SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 7,2014	SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 6,2015	SEM 8,2014	
Kamis	07.30-10.00	ARSITI [A]	ARSITI [C]	IAK [A]	KAI [A]	KAI [C]	KAI [D]	PKETI [A]			
		SEM 2,2017	SEM 2,2017	SEM 8,2014	SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 7,2014			

Hari	Jam	Mata Kuliah									
	10.15-12.45	KAI [C]	KAI [D]	KKI [C]	MRTI [B]	MRPHP [C]	PBW [A]	PSO [A]	TTI [A]		
		SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 4,2016	SEM 2,2017	SEM 6,2015		
	13.30-16.00	EB IS	KAI [A]	KKI [D]	PBW [B]	PBW [D]	TKTI [B]				
		SEM 8,2014	SEM 4,2016	SEM 2,2017	SEM 4,2016	SEM 4,2016	SEM 6,2015				
Jumat	13.00-15.30	AUDIT SI	IMK [C]	KAI [B]	MA BD [C]	MAB D [D]	MPITI [B]				
		SEM 7,2014	SEM 4,2016	SEM 4,2016	SEM 6,2015	SEM 6,2015	SEM 6,2015				

LAMPIRAN E : Hasil Optimasi Jadwal Ganjil

Hari	Jam	MATKUL										
Senin	07.30-10.00	DMJK [B]	MATDIS [A]	MATDIS [B]	MATDIS [C]	MPTI [C]	PBO [D]	PKETI [A]	PKETI [B]	PKETI [C]	RO [A]	
		3 sks/ sem 3	3 sks/ sem 1	3 sks/ sem 1	3 sks/ sem 1	4 sks/ sem 5	3 sks/ sem 3	3 sks/ sem 7	3 sks/ sem 7	3 sks/ sem 7	3 sks/ sem 5	
	10.15-12.45	BP [A]	BP [C]	DMPB [A]	PKB [A]	PSDP [A]	PSDP [B]	PSSI [B]	RO [C]	RO [D]	SS [A]	SS [B]
		4 sks/ sem 1	4 sks/ sem 1	4 sks/ sem 3	3 sks/ sem 7	4 sks/ sem 5	4 sks/ sem 5	3 sks/ sem 7	3 sks/ sem 5	3 sks/ sem 5	2 sks/ sem 5	2 sks/ sem 5
	13.30-16.00	PBO [C]	PSDP [D]	STAT [A]								
		3 sks/ sem 3	4 sks/ sem 5	4 sks/ sem 3								

Hari	Jam	MATKUL										
Selasa	07.30-10.00	KCB [A]	KCB [B]	KCB [D]	MO [B]	PSSI [C]	PSSI [E]	RO [B]	STI [D]			
		3 sks/sem 7	3 sks/sem 7	3 sks/sem 7	3 sks/sem 1	3 sks/sem 7	3 sks/sem 7	3 sks/sem 5	3 sks/sem 1			
	10.15-12.45	KCB [C]	KPPL [B]	MO [D]	MPTI [A]	MPTI [C]	MPTI [D]	PSSI [A]	PSSI [D]	STAT [B]	STI [A]	STI [C]
		3 sks/sem 7	3 sks/sem 5	3 sks/sem 1	4 sks/sem 5	4 sks/sem 5	4 sks/sem 5	3 sks/sem 7	3 sks/sem 7	4 sks/sem 3	3 sks/sem 1	3 sks/sem 1
	13.30-16.00	SS [C]	STI [B]									
		2 sks/sem 5	3 sks/sem 1									
Rabu	07.30-10.00	DDPL [A]	DMJK [D]	DMPB [C]	MLTI [A]	MLTI [D]	MPITI [A]	PSDP [B]	SC [A]	SPK [A]	SPK [B]	TEKPER [A]

Hari	Jam	MATKUL										
		3 sks/sem 3	3 sks/sem 3	4 sks/sem 3	3 sks/sem 5	3 sks/sem 5	3 sks/sem 6	4 sks/sem 5	3 sks/sem 6	3 sks/sem 7	3 sks/sem 7	3 sks/ sem 7
	10.15-12.45	AUDIT [A]	BP [D]	DMJK [C]	DMPB [A]	E-BIS [A]	FORDIG [A]	KPPL [C]	PBO [B]	PDAB [A]	PSDP [D]	STAT [B]
		3 sks/sem 7	4 sks/sem 1	3 sks/sem 3	4 sks/sem 3	3 sks/sem 7	3 sks/ sem 7	3 sks/sem 5	3 sks/sem 3	3 sks/sem 7	4 sks/sem 5	4 sks/sem 3
	13.30-16.00	TOST [A]										
		3 sks/sem 7										
Kamis	07.30-10.00	DDPL [B]	DDPL [C]	EP [B]	EP [Q]	KPPL [A]	KPPL [D]	MATDIS [D]	MLTI [B]	MPTI [B]	PBO [A]	PSDP [C]
		3 sks/ sem 3	3 sks/ sem 3	2 sks/ sem 7	2 sks/sem 7	3 sks/sem 5	3 sks/sem 5	3 sks/sem 1	3 sks/sem 5	4 sks/sem 5	3 sks/sem 3	4 sks/sem 5

Hari	Jam	MATKUL										
	10.15-12.45	DMPB [B]	DMPB [D]	EP [A]	KCB [E]	MRTI [A]	MLTI [C]	MPTI [A]	MPTI [D]	PBD [D]	STAT [A]	
		4 sks/sem 3	4 sks/sem 3	2 sks/sem 7	3 sks/sem 7	3 sks/sem 6	3 sks/sem 5	4 sks/sem 5	4 sks/sem 5	3 sks/sem 3	4 sks/sem 3	
	13.30-16.00	TTI [A]										
		3 sks/sem 7										
Jumat	07.30-10.00	BP [A]	BP [B]	BP [C]	BP [D]	KAI [A]	MO [A]	MPTI [B]	PBD [A]	PBD [B]	PBD [C]	PSDP [C]
		4 sks/sem 1	4 sks/sem 1	4 sks/sem 1	4 sks/sem 1	4 sks/sem 4	3 sks/sem 1	4 sks/sem 5	3 sks/sem 3	3 sks/sem 3	3 sks/sem 3	4 sks/sem 5
	13.00 - 15.30	ASD [A]	BP [B]	DMJK [A]	DMPB [B]	DMPB [C]	DMPB [D]	KAI [A]	MO [C]	PSDP [A]		

Hari	Jam	MATKUL										
		3 sks/sem 2	4 sks/sem 1	3 sks/sem 3	4 sks/sem 3	4 sks/sem 3	4 sks/sem 3	4 sks/sem 4	3 sks/sem 1	4 sks/sem 5		

LAMPIRAN F : Hasil Optimasi Jadwal Genap

Hari	Jam	MATKUL										
Senin	07.30-10.00	ADPL [A]	DBD [D]	DMPB [A]	IAK [A]	MRTI [D]	MPITI [A]	MPITI [B]	PBO	PBW [D]	PKETI [A]	PSO [C]
		SEM 4,2016	SEM 4,2016	SEM 3,2016	SEM 8,2014	SEM 6,2015	SEM 6,2015	SEM 6,2015	SEM 3,2016	SEM 4,2016	SEM 7,2014	SEM 2,2017
	10.15-12.45	ADPL [C]	ASD [C]	DBD [C]	DMPB [A]	IMK [A]	MRPHP [A]	MRPHP [B]	MRP HP [C]	PINT [A]	PSO [B]	ROL [A]
		SEM 4,2016	SEM 2,2017	SEM 4,2016	SEM 3,2016	SEM 4,2016	SEM 6,2015	SEM 6,2015	SEM 6,2015	SEM 8, 2014	SEM 2,2017	SEM 8,2014
	13.30-16.00	PSO [D]	STAT [B]									
		SEM 2,2017	SEM 3,2016									
Selasa	07.30-10.00	EP [A]	IMK [B]	IMK [D]	KKI [A]	KKI [C]	KKI [D]	KPPL [A]	MAB D [A]	MABD [B]	MRTI [C]	MLTI [A]
		SEM 7,2014	SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 2,2017	SEM 2,2017	SEM 7,2014	SEM 6,2015	SEM 6,2015	SEM 6,2015	SEM 5,2015

Hari	Jam	MATKUL										
	10.15-12.45	ADPL [B]	ADPL [D]	ARSITI [B]	ASD [A]	KKI [B]	MRTI [A]	MRPHP [D]	PBW [C]	PDAB [A]	PPB [A]	STAT [A]
		SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 2,2017	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 4,2016	SEM 8,2014	SEM 8,2014	SEM 3,2016
	13.30-16.00	ADPL [C]	TTI [B]	TTI [D]								
		SEM 4,2016	SEM 6,2015	SEM 6,2015								
Rabu	07.30-10.00	ADPL [A]	ASD [D]	BAPRO [A]	DBD [A]	DBD [B]	DBD [C]	DBD [D]	EP [Q]	KAI [B]	MATDIS [A]	MKTI [A]
		SEM 4,2016	SEM 2,2017	SEM 1,2017	SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 7,2014	SEM 4,2016	SEM 1,2017	SEM 8,2014
	10.15-12.45	ADPL [B]	ADPL [D]	ARSITI [C]	ARSITI [D]	AUDITSI	IMK [C]	KAI [A]	KAI [D]	KKI [A]	KKI [B]	MPITI [C]
		SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 2,2017	SEM 7,2014	SEM 4,2016	SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 2,2017	SEM 6,2015
	13.30 - 16.00	STAT [B]	TEKPER [A]									

Hari	Jam	MATKUL										
		SEM 3,2016	SEM 8,2014									
Kamis	07.30-10.00	ARSITI [A]	KAI [A]	KAI [C]	KKI [C]	KKI [D]	MABD [D]	SC [A]	SC [B]	SPK [A]	STAT [A]	TKTI [A]
		SEM 2,2017	SEM 4,2016	SEM 4,2016	SEM 2,2017	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 6,2015	SEM 8,2014	SEM 3,2016	SEM 6,2015
	10.15-12.45	DBD [B]	KAI [D]	MRTI [B]	PBW [A]	PSO [A]	SC [C]	SC [D]	SPK [B]	TKTI [B]	TTI [A]	TTI [C]
		SEM 4,2016	SEM 4,2016	SEM 6,2015	SEM 4,2016	SEM 2,2017	SEM 6,2015	SEM 6,2015	SEM 8,2014	SEM 6,2015	SEM 6,2015	SEM 6,2015
	13.30-16.00	TOST [A]										
		SEM 8,2014										
Jumat	13.00-15.30	ASD [B]	BAPRO [A]	DBD [A]	EBIS	KAI [B]	KAI [C]	MABD [C]	PBW [B]	TKTI [C]	TKTI [D]	
		SEM 2,2017	SEM 1,2017	SEM 4,2016	SEM 8,2014	SEM 4,2016	SEM 4,2016	SEM 6,2015	SEM 4,2016	SEM 6,2015	SEM 6,2015	

Halaman ini sengaja dikosongkan