TESIS - IS185401

PENYELESAIAN PENJADWALAN MATA KULIAH MENGGUNAKAN METODE HIPERHEURISTIK DENGAN HIBRIDISASI ALGORITMA *TABU SEARCH*, *SIMULATED ANNEALING*, DAN *SELF-ADAPTIVE* PADA LINTAS DOMAIN PERMASALAHAN

KARTIKA MAULIDA HINDRAYANI
NRP. 05211750010005

DOSEN PEMBIMBING:
AHMAD MUKLASON, S.Kom., M.Sc., Ph.D.
NIP. 198203022009121009

Departemen Sistem Informasi
Fakultas Teknologi Elektro Dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
2020

Halaman ini sengaja dikosongkan

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom)
di
Institut Teknologi Sepuluh Nopember
Oleh:

KARTIKA MAULIDA HINDRAYANI
NRP: 05211750010005

Tanggal Ujian: 03 Januari 2020
Periode Wisuda: Maret 2020

Disetujui oleh:
Pembimbing:

1. Ahmad Muklason, S.Kom., M.Sc., Ph.D.
   NIP: 19820302 200912 1 009 .....

Penguji:

1. Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D. .....
   NIP: 19820120 200501 2 001

2. Faizal Mahananto, S.Kom., M.Eng, Ph.D.
   NIP: 19851031 201903 1 009 .....

Kepala Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas

Dr. Mudjahidin, S.T., M.T.
NIP: 19701010 200312 1 001

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Alhamdulillahi robbil ‘alamin, segala puji bagi Allah SWT atas limpahan nikmat, rahmat, petunjuk, dan karunia-Nya, sehingga penulis dapat menyelesaikan laporan Tesis dengan judul, “Penyelesaian Penjadwalan Mata Kuliah Menggunakan Metode Hiperheuristik Dengan Hibridisasi Algoritma Tabu Search, Simulated Annealing, Dan Self-Adaptive Pada Lintas Domain Permasalahan” yang menjadi salah satu syarat kelulusan pada Departemen Sistem Informasi, Fakultas Teknologi Elektro dan Informatika Cerdas, Institut Teknologi Sepuluh Nopember Surabaya. Dalam penyelesaian laporan ini, penulis telah banyak mendapat bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Mudjahidin, S.T., M.T., selaku Kepala Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
2. Bapak Ahmad Mukhlason, S.Kom., M.Sc., Ph.D. selaku Pembimbing Tesis yang telah meluangkan waktu untuk memberikan bimbingan selama proses pembuatan laporan ini.
3. Ibu Nur Aini Rachmawati, S.Kom., M.Sc., Ph.D. dan Bapak Faizal Mahananto, S.Kom., M.Eng., Ph.D selaku dosen penguji sidang Tesis.
4. Pihak-pihak terkait yang telah membantu hingga selesainya penyusunan laporan Tesis ini.

Penulis menyadari bahwa laporan Tesis ini masih banyak kekurangan. Untuk itu, penulis mohon maaf jika ada kesalahan dan kekurangan dalam penyusunan laporan ini. Kritikan dan saran dari pembaca yang bersifat membangun dapat dikirimkan melalui email kartikamaulida@gmail.com. Semoga laporan ini dapat memberikan manfaat bagi pembaca dan penulis.

Surabaya, Januari 2020

Penulis

Halaman ini sengaja dikosongkan

Penyelesaian Penjadwalan Mata Kuliah Menggunakan Metode Hiperheuristik Dengan Hibridisasi Algoritma *Tabu Search, Simulated Annealing,* Dan *Self-Adaptive* Pada Lintas Domain Permasalahan

Nama mahasiswa : Kartika Maulida Hindrayani

NRP : 05211750010005

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

ABSTRAK

Penjadwalan diperlukan sebagai pengalokasian sumber daya untuk menyelesaikan sebuah pekerjaan dengan batasan-batasan yang telah didefinisikan sehingga dapat memaksimalkan kemungkinan alokasi atau meminimalisir pelanggaran batasan. Salah satu jenis penjadwalan pada bidang pendidikan yaitu *Post-Enrollment Course Timetabling (PE-CTT)*. Tantangan yang dihadapi pada PE-CTT yaitu perbedaan permasalahan, sejumlah batasan, dan persyaratan berbeda pada satu universitas dengan universitas lainnya sehingga sulit untuk menemukan solusi yang umum dan efektif. Salah satu solusi yang dapat mengembangkan sistem yang lebih *general* dengan menggunakan metode yang lebih murah dan tetap dapat menyelesaikan masalah adalah dengan menggunakan pendekatan *Hyper-Heuristic*. Pengujian akan dilakukan pada lintas domain yaitu dataset Socha dan dataset ITC-2007. Strategi *Self-Adaptive* digunakan sebagai strategi untuk memilih *Low-Level-Heuristic* (LLH) dan *Simulated Annealing* dan *Tabu Search* sebagai strategi *Move Acceptance* (MA) untuk menyelesaikan permasalahan penjadwalan mata kuliah tersebut.

Hasil yang didapatkan pada dataset Socha, algoritma SATSSA menghasilkan nilai yg lebih baik dibandingkan dengan algoritma lain pada *2 instance*. Algoritma SATSSA mampu mencapai nilai yang paling optimum pada 5 *instance* dataset Socha. Algoritma SATSSA menghasilkan nilai yang lebih optimum dibandingkan dengan algoritma lain pada 5 instance dataset ITC2007 yaitu instance early1, early2, hidden20, hidden21, dan hidden23.

Kata Kunci : *Post Enrollment Course Timetabling, Simulated Annealing, Tabu Search, Self-Adaptive,* Dataset Socha, Dataset ITC-2007

Halaman ini sengaja dikosongkan

Solving Course Timetabling Using Hyperheuristic Method with Hybridization of Taboo Search, Simulated Annealing, and Self-Adaptive Algorithm in Cross Domain Problems

By	: Kartika Maulida Hindrayani
Student Identity Number	: 05211750010005
Supervisor	: Ahmad Muklason, S.Kom , M.Sc., Ph.D

ABSTRACT

Timetabling is needed to complete a job with allocating resources and defined boundaries to maximize the possibility of allocation or minimize the violation of boundaries. One type of timetabling in the education field is Post-Enrollment Course Timetabling (PE-CTT). The challenges faced in the PE-CTT are differences in problems, a number of limitations, and requirements that differ from one university to another so that it is difficult to find common and effective solutions. One solution that can develop more general systems by using cheaper methods and still being able to solve problems is the Hyper-Heuristic approach. Testing will be carried out on cross domains namely the Socha dataset and the ITC-2007 dataset. The Self-Adaptive Strategy is used as a strategy for selecting Low-Level-Heuristics (LLH) and Simulated Annealing and Taboo Search as a Move Acceptance (MA) strategy to solve the course timetabling problems.

The results obtained in the Socha dataset, SATSSA algorithm produces better values compared to other algorithms in 2 instances. SATSSA algorithm is able to achieve the most optimum value on 5 Socha dataset instances. SATSSA algorithm produces more optimum values compared to other algorithms on 5 ITC2007 dataset instances, namely early1, early2, hidden20, hidden21, and hidden23 instances.

Key words : Post Enrollment Course Timetabling, Simulated Annealing, Tabu Search, Self-Adaptive, Socha Dataset, ITC-2007 Dataset

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR PSEUDOCODE

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Bab ini terdiri dari latar belakang dilakukannya penelitian, perumusan masalah, tujuan dan kontribusi penelitian, batasan penelitian, dan sistematika penulisan.

1.1 Latar Belakang

Salah satu topik yang sering dibahas pada riset operasional adalah penjadwalan. Hal ini menunjukkan perlunya penjadwalan, yaitu pengalokasian sumber daya untuk menyelesaikan sebuah pekerjaan dengan batasan-batasan yang telah didefinisikan sehingga dapat memaksimalkan kemungkinan alokasi atau meminimalisir pelanggaran batasan tersebut. Definisi lain yaitu penjadwalan merupakan alokasi obyek untuk ditempatkan sesuai waktu dan batasan yang diberikan sehingga dapat memenuhi sedekat mungkin kepada tujuan yang diinginkan [1]. Solusi yang layak untuk permasalahan penjadwalan biasanya merupakan solusi yang memenuhi semua persyaratan *hard constraints*. *Hard constraints* merupakan kondisi yang harus dipenuhi, sedangkan *soft constraints* kondisi yang ingin dipenuhi tetapi tidak begitu penting [2]. Pelanggaran *hard constraints* dapat menjadikan sebuah solusi tidak layak, sementara pelanggaran *soft constraints* dapat dikenakan nilai penalti.

Penjadwalan dapat diaplikasikan pada bidang pendidikan, olahraga, transportasi, pekerjaan oleh mesin atau karyawan, dsb. Pada penelitian ini akan dikhususkan membahas penjadwalan pada bidang pendidikan. Penjadwalan pada bidang pendidikan sangat penting dalam penyelenggaran operasional di perguruan tinggi. Penjadwalan pada bidang pendidikan dapat dibagi dalam penjadwalan sekolah, penjadwalan ujian, dan penjadwalan mata kuliah. Penjadwalan mata kuliah pada universitas meliputi bagaimana menetapkan sejumlah *event* (yaitu perkuliahan, seminar, lab, tutorial, dll) ke dalam timeslot dan ruangan yang memiliki batasan [3].

Penelitian ini akan membahas permasalahan UCTP (*University Course Timetabling Problem*). UCTP dapat didefinisikan sebagai pengalokasian ruangan yang terbatas, mata kuliah, mahasiswa dan dosen pada batas waktu dan memenuhi batasan yang telah ditetapkan [4]. Topik yang dibahas pada UCTP biasanya ada dua jenis yaitu *Post-Enrolment Course Timetabling* (PE-CTT), *Curriculum Based Course Timetabling* (CB-CTT), dan *Examination Timetabling* (ETP). Perbedaan mendasar PE-CTT dengan CB-CTT adalah penjadwalan kuliah untuk mata kuliah dalam jumlah ruangan dan periode waktu tertentu, di mana konflik antar mata kuliah diatur sesuai dengan kurikulum yang ditetapkan oleh universitas dan bukan diatur sesuai pendaftaran mata kuliah oleh mahasiswa [5]. Permasalahan UCTP merupakan permasalahan kombinatorial *NP-Hard*. Hal tersebut dikarenakan UCTP memiliki perbedaan permasalahan, sejumlah batasan, dan persyaratan berbeda pada satu universitas dengan universitas lainnya sehingga sulit untuk menemukan solusi yang umum dan efektif. Perbedaan kondisi universitas satu dengan universitas lainnya menghasilkan sistem penjadwalan universitas yang sukses digunakan hanya pada masing-masing universitas [6].

Pendekatan-pendekatan yang dapat digunakan untuk menyelesaikan permasalahan penjadwalan yaitu pendekatan *heuristic, metaheuristic, hybrids, constraint-based methods, hyper-heuristics*, dll. Algoritma *metaheuristics* yang sering digunakan, membutuhkan parameter tuning untuk setiap permasalahan. Salah satu solusi yang dapat mengembangkan sistem yang lebih *general* dengan menggunakan metode yang lebih murah dan tetap dapat menyelesaikan masalah adalah dengan menggunakan pendekatan *Hyper-Heuristic*. *Hyper-Heuristic* merupakan metode sederhana yang melakukan pencarian pada search space heuristic [2]. Definisi dari *Hyper-Heuristic* adalah heuristik yang memilih heuristik atau heuristik yang menghasilkan heuristik.

Burke mempublikasikan konsep *Simulated Annealing HyperHeuristic* pada kasus penjadwalan mata kuliah [3]. Terdapat dua batasan pokok pada permasalahan UCTP yaitu tidak ada siswa yang dijadwalkan di dua *event* pada

timeslot yang sama, selain itu kapasitas dan fitur ruangan harus memenuhi persyaratan event.

Leng Goh menyelesaikan permasalahan PE-CTT dengan menggunakan algoritma *Tabu Search with Sampling and Perturbation* (TSSP) pada tahap pertama kemudian pada tahap kedua menggunakan *Simulated Annealing with Reheating* (SAR) untuk meningkatkan solusi [7]. Algoritma mengevaluasi *non-tabu* timeslot untuk sampled *events* tertentu, jumlah *events* tersebut acak, solusi kandidat dengan angka paling kecil pada *event* yang belum diletakkan lebih disukai. Kemudian solusi saat ini "diganggu" pada interval iterasi tertentu dan event diletakkan pada setiap timeslot yang kosong pada slotList untuk diacak dengan menggunakan Swap atau operator Kempe. Namun kekurangan dari tahap pertama adalah apabila diiterasi terlalu sering dapat memperlambat pencarian solusi yang layak. SA merupakan algoritma yang efektif dalam menyelesaikan masalah optimisasi kombinatorial. Penentuan suhu di SA digunakan untuk menerima gerakan yang menanjak. Apabila pencarian masih tetap setelah dilakukan pemanasan sebelumnya maka suhu yang lebih tinggi digunakan untuk pemanasan selanjutnya. Hasil yang kompetitif didapatkan dari penelitian tersebut.

Pan menyelesaikan permasalahan penjadwalan *lot-streaming flow shop* dengan menggunakan algoritma *Artificial Bee Colony* yang dikombinasikan dengan strategi *self-adaptive* [8]. Penerapan strategi *self-adaptive* pada algoritma *Artificial Bee Colony* tersebut diharapkan dapat mengeksplorasi solusi yang menjanjikan di *search space*. Hasil dari algoritma tersebut adalah efektifitas dan efisiensi pada performa komputasional dibanding algoritma lainnya. Menurut Liu *Self-adaptive* dapat meningkatkan tingkat konvergensi ke nilai optimal pada keseluruhan proses optimisasi [9]. Liu et al menggunakan strategi *self-adaptive* pada algoritma *Artificial Bee Colony* agar dapat mengarahkan setiap individu lebah ke nilai yang optimal.

Ahsanul (2018) mengembangkan sebuah program dengan permasalahan PE-CTT menggunakan *Tabu Search* dan *Simulated Annealing*. Dataset yang digunakan adalah Socha. Penerapan komputasi dengan dataset yang lebih general

akan dilakukan pada penelitian ini sehingga memerlukan dataset lain untuk digunakan. Dataset yang dipilih yaitu ITC-2007. Beberapa tantangan yang ditemukan dalam uji coba awal penerapan dataset ITC 2007, yaitu solusi awal yang membutuhkan waktu lama, *soft constraints* yang berbeda, dan tentu saja algoritma yang dapat menghasilkan solusi yang lebih optimal. Strategi *Self-Adaptive* yang memiliki pengetahuan mengenai nilai-nilai sebelumnya diharapkan dapat menghasilkan solusi yang lebih baik dan meningkatkan tingkat konvergensi ke nilai optimal. Struktur dari *Self-Adaptive* yang sederhana dan memiliki kinerja dinamis dapat meningkatkan kemampuan *search algorithm*.

Pada penelitian ini akan menyelesaikan permasalahan PE-CTT dengan menggabungkan algoritma *Self-Adaptive, Simulated Annealing,* dan *Tabu Search*. Dataset yang digunakan adalah dataset Socha dan ITC 2007. Penelitian ini bertujuan untuk mengevaluasi metode yang diusulkan dengan pengujian terhadap dataset Socha dan ITC 2007.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, diketahui bahwa terdapat permasalahan dalam menyelesaikan PE-CTT yaitu bagaimana mengalokasikan ruangan yang terbatas, mata kuliah, mahasiswa dan dosen pada batas waktu dan memenuhi batasan yang telah ditetapkan di mana tiap universitas dapat berbeda-beda. Penelitian sebelumnya telah menunjukkan bahwa kombinasi *Simulated Annealing dengan Tabu Search,* dan *Artificial Bee Colony* dengan strategi *Self-Adaptive* menghasilkan solusi yang kompetitif, sehingga rumusan masalah yang ingin dijawab yaitu :

1. Bagaimana struktur metode hyper-heuristic pada gabungan dari Algoritma *Self-Adaptive, Simulated Annealing,* dan *Tabu Search*?
2. Bagaimana performa dari algoritma yang diusulkan?

1.3 Tujuan dan Manfaat Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah menyelesaikan permasalahan *Post-Enrolment Course Timetabling* dengan benchmark dataset ITC-2007 dan Dataset Socha agar dihasilkan solusi penjadwalan dengan nilai pelanggaran yang minimal. Permasalahan tersebut akan diselesaikan dengan penerapan algoritma *Simulated Annealing, Tabu Search,* dan *Self-Adaptive* pada aplikasi yang telah dibuat dalam ruang lingkup *Hyper-Heuristic*.

Manfaat dari penelitian ini adalah sebuah referensi untuk penerapan metode algoritma *Simulated Annealing, Tabu Search,* dan *Self-Adaptive* pada ruang lingkup *Hyper-Heuristic*. Penjadwalan dengan nilai penalti *soft-constraints* yang minimal juga termasuk manfaat dari penelitian sehingga dapat mendukung operasional universitas.

1.4 Kontribusi Penelitian

Hasil dari penelitian ini adalah sebuah strategi *high level heuristic* dalam hiperheuristik yang diharapkan dapat memberikan kontribusi secara teoritis maupun secara praktis. Kontribusi teoritis maupun praktis dibahas pada sub bab berikut

1.4.1. Kontribusi Teoritis

Kontribusi teoritis dari penelitian ini yaitu kombinasi algoritma *Tabu Search*, *Simulated Annealing* dan *Self-Adaptive* dalam kerangka kerja hiperheuristik untuk menyelasaikan permasalahan penjadwalan mata kuliah agar solusi dapat lebih baik. Penyajian evaluasi kinerja dari hibridisasi ketiga algoritma tersebut juga menjadi kontribusi teoritis. Kontribusi dari penelitian ini didapatkan dari pengujian terhadap adaptasi algoritma *Self-Adaptive* (SA) sebagai mekanisme seleksi LLH dalam mencari solusi optimal. Kemudian SA dikombinasikan dengan metode Tabu Search (TS) dan Simulated Annealing (SA) sebagai *move acceptance* sehingga menjadi strategi *high level heuristic* baru dalam menyelesaikan masalah optimasi lintas domain pada kerangka kerja hiperheuristik.

1.4.2. Kontribusi Praktis

Kontribusi praktis dari penelitian ini adalah Otomasi proses penjadwalan dan optimasi proses penjadwalan sehingga proses penjadwalan mata kuliah yang digunakan di perguruan tinggi menjadi lebih efektif.

1.5 Batasan Penelitian

Penelitian ini memiliki ruang lingkup yang akan menjadi batasan dalam penelitian ini. Batasan penelitian ini antara lain:

1. Dataset yang digunakan adalah dataset Socha dan dataset ITC 2007
2. Permasalahan yang akan diselesaikan adalah *Post Enrolment Course Timetabling*

1.6 Sistematika Penulisan

Sistematika penulisan penelitian tesis ini adalah sebagai berikut :

1. **Bab I: Pendahuluan**

   Bab ini berisikan pendahuluan yang menjelaskan latar belakang permasalahan, perumusan masalah, tujuan penelitian, manfaat penelitian, kontribusi penelitian, batasan penelitian serta sistematika penulisan.

2. **Bab II: Kajian Pustaka**

   Bab ini berisikan kajian terhadap teori dan penelitian-penelitian yang sudah ada sebelumnya. Kajian pustaka ini bertujuan untuk memperkuat dasar dan alasan dilakukan penelitian.

3. **Bab III: Metodologi Penelitian**

   Bab ini berisikan mengenai rancangan penelitian, lokasi dan tempat penelitian, serta tahapan-tahapan sistematis yang digunakan selama melakukan penelitian.

4. **Bab IV: Riset Pendahuluan**

   Bab ini berisikan mengenai penjelasan riset pendahuluan yang telah dilakukan sebelumnya, performa dari riset pendahuluan tersebut, dan pengembangan penelitian yang akan dilakukan.

5. **Bab V:Perancangan**

   Bab ini berisikan mengenai hasil rancangan yang dibutuhkan dalam melaksanakan penelitian.

6. **Bab VI Hasil dan Pembahasan**

   Bab ini berisikan mengenai hasil yang diperoleh selama pelaksanaan penelitian serta membahas hasil tersebut.

7. **Daftar Pustaka**

   Bagian ini berisikan daftar referensi yang digunakan dalam penelitian ini, baik jurnal, buku, maupun artikel.

Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Bab ini menjelaskan mengenai teori-teori yang digunakan dalam penyusunan tesis serta kajian pustaka yang diambil dari penelitian-penelitian sebelumnya yang relevan. Kajian pustaka ini selanjutnya akan dibangun sebagai landasan dalam melakukan penelitian ini.

2.1.Penelitian Terkait

Pada bagian ini akan dijelaskan mengenai beberapa penelitian yang terkait dengan penelitian yang akan dilakukan. Penelitian-penelitian yang akan dibahas merupakan beberapa penelitian mengenai *Post Enrolment-Course Timetabling* (PE-CTT) dan *Curriculum Based Course Timetabling* (CB-CTT).

2.1.1. Penelitian Sebelumnya

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

Tabel 2.1 Penelitian Terkait

Peneliti	Dataset	Metode	Deskripsi	Gap
[3]		*Simulated Annealing Hyper-Heuristics*	Peneliti mengusulkan pendekatan *simulated annealing Hyper-Heuristics* yang mengelola serangkaian *neighbourhood functions* atau heuristik dan secara dinamis memilih heuristik tersebut.	Pada penelitian tersebut tidak menguji coba dataset, hanya usulan pendekatan *Simulated Annealing Hyper-Heuristics.* Sementara penelitian mendatang menggunakan gabungan *Tabu Search, Simulated Annealing,* dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.

Peneliti	Dataset	Metode	Deskripsi	Gap
[10]	ITC-2011	*Greedy Gradient Selection Heuristic* dikombinasikan dengan *Simulated Annealing Move Acceptance*	Eksperimen menunjukkan bahwa *Greedy Gradient* jika dikombinasikan dengan *Simulated Annealing Move Acceptance* mampu menghasilkan solusi yang lebih baik dari heuristic lainnya.	Pada penelitian tersebut menguji coba dataset ITC-2011 dengan menggunakan metode *Greedy Gradient* dan *Simulated Annealing*. Sementara penelitian mendatang akan menguji coba dataset Socha dan dataset ITC-2007 menggunakan gabungan *Tabu Search*, *Simulated Annealing*, dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.
[11]		*Tabu Search Hyper-Heuristics*	Membandingkan kinerja *Tabu Search Hyper-Heuristics* dengan *Simple Low Level Heuristics (add, drop, swap)* dan algoritma *Ant*. Hasilnya *Hyper-Heuristics* terbukti dapat memberikan solusi yang layak dan kompetitif.	Penelitian mendatang akan berfokus pada permasalahan penjadwalan mata kuliah universitas menggunakan gabungan *Tabu Search*, *Simulated Annealing*, dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.
[12]	ITC-2007	Algoritma Genetika dan *Tabu Search*	Mengkombinasikan metode Hybrid Algoritma Genetika dan *Tabu Search* pada domain PE-CTT. Hasilnya memberikan solusi yang kompetitif dan	Pada penelitian tersebut menggunakan Algoritma Genetika dan *Tabu Search* . Sementara penelitian mendatang menggunakan

Peneliti	Dataset	Metode	Deskripsi	Gap
			dapat bekerja dengan baik.	gabungan *Tabu Search, Simulated Annealing,* dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.
[13]	ITC-2007, Lewis and Paechter, ITC-2002, *Metaheuristics Network*	*Simulated Annealing*	Peneliti menggunakan algoritma Simulated Annealing untuk menyelesaikan permasalahan 4 dataset. Initial Solution yang digunakan adalah Greedy dan Greedy dengan batasan waktu	Penelitian tersebut menguji coba 4 dataset dan tidak ada dataset Socha. Sementara penelitian mendatang akan menguji coba dataset Socha dan dataset ITC-2007 menggunakan gabungan *Tabu Search, Simulated Annealing,* dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.
Ahsanul	Socha	*Simulated Annealing – Tabu Search Hyper-Heuristic*	Solusi awal diperoleh dengan menggunakan algoritma *Greedy*. Kemudian LLH yang digunakan adalah *swap* dan *move*. Proses optimasi menggunakan SA dan Tabu Search Hyper-Heuristics. Hasilnya dapat memberikan hasil yang optimum.	Penelitian tersebut hanya menguji coba dataset Socha. Sementara penelitian mendatang akan menguji coba dataset Socha dan dataset ITC-2007 menggunakan gabungan *Tabu Search, Simulated Annealing,* dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.
[9]	Reuters-	*Artificial*	Pengkombinasian	Pada penelitian

Peneliti	Dataset	Metode	Deskripsi	Gap
	21,578, Classic4, dan WebKB	*Bee Colony, Self-Adaptive Strategy*	*Artificial Bee Colony* dengan strategi *Self-Adaptive* dalam penyelesaian berbagai dataset. Hasil dari metode tersebut menunjukkan efektif disebabkan meningkatnya kemampuan optimasi algoritma dengan peningkatan tingkat konvergensi algoritma.	tersebut tidak menyelesaikan permasalahan *Post-Enrollment Course Timetabling*. Sementara penelitian mendatang akan menguji coba dataset Socha dan dataset ITC-2007 menggunakan gabungan *Tabu Search, Simulated Annealing,* dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.
[8]		*Artificial Bee Colony, Self-Adaptive Strategy*	Pengkombinasian *Artificial Bee Colony* dengan strategi *Self-Adaptive* dalam penyelesaian permasalahan *lot-streaming flow shop*. Hasil dari metode tersebut menunjukkan solusi kompetitif, efektif dan efisien pada penerapan komputasi	Pada penelitian tersebut menguji coba mengenai lot-streaming flow shop. Sementara penelitian mendatang akan menguji coba dataset Socha dan dataset ITC-2007 menggunakan gabungan *Tabu Search, Simulated Annealing,* dan *Self-Adaptive* secara gabungan dalam *framework Hyper-heuristics*.

2.2. Dasar Teori

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

2.2.1. *Educational Timetabling Problem*

Permasalahan *timetabling* dapat didefinisikan sebagai permasalahan meletakkan sejumlah *events* ke dalam periode waktu yang terbatas. Wren (1996) mendefinisikan *timetabling* sebagai pengalokasian sumber daya yang diberikan ke dalam waktu dan batasan yang ditentukan sehingga dapat sedekat mungkin pada tujuan yang diinginkan. Terdapat dua kategori batasan yaitu *hard constraint* dan *soft constraint*. *Hard constraint* merupakan batasan yang harus dipenuhi sehingga solusinya layak, sedangkan *soft constraint* merupakan kondisi yang diinginkan dan diperbolehkan dilanggar namun memiliki nilai penalti [2]. Permasalahan *timetabling* telah diaplikasikan pada bidang pendidikan, olahraga, transportasi, karyawan. Jenis penjadwalan pada bidang pendidikan yaitu *examination timetabling* dan *course timetabling*. Perbedaan *examination timetabling* dengan *course timetabling* adalah pada ruangan dan periode waktu [6]. Pada *examination timetabling,* beberapa ujian dapat dijadwalkan pada ruangan yang sama sedangkan pada *course timetabling* tidak bisa. Pada *course timetabling*, mahasiswa dapat mengikuti dua mata kuliah atau lebih dalam sehari sementara pada *exam timetabling* hal tersebut sebisa mungkin diminimalisir.

Pada bidang pendidikan, *hard constraints* biasanya adalah [4]:

- tidak ada mahasiswa yang mengambil lebih dari satu kelas pada waktu yang sama
- hanya ada satu kelas pada satu ruangan di satu waktu
- tidak ada dosen yang mengajar lebih dari satu kelas pada waktu yang sama

Soft constraints biasanya adalah [6]:

- Penempatan waktu. Penjadwalan sebuah mata kuliah/ujian bisa saja harus ditempatkan pada waktu tertentu
- Batasan waktu di antara events. Penjadwalan waktu mata kuliah/ujian sebelum atau sesudah mata kuliah/ujian lainnya

- Penyebaran event dalam jangka waktu tertentu. Mahasiswa tidak boleh mendapatkan ujian pada waktu yang berurutan atau dua ujian pada hari yang sama
- Kecocokan. Dosen bisa saja menginginkan mata kuliahnya pada hari tertentu. Batasan ini dapat berbenturan dengan batasan penyebaran event.
- Penempatan sumber daya. Dosen menginginkan mengajar pada ruangan tertentu sehingga mata kuliah/ujian harus dijadwalkan pada ruangan tertentu

Besarnya jumlah event dan batasan yang bermacam-macam mengakibatkan keseluruhan kumpulan solusi sangat banyak. Hal tersebut menjadikan penjadwalan membutuhkan usaha yang banyak apabila dikerjakan secara manual, sehingga para peneliti tertarik mengembangkan penjadwalan menggunakan komputasi. Beberapa tipe pendekatan untuk menyelesaikan masalah penjadwalan menurut Burke (2002) adalah metode sequential (menggunakan domain heuristic), metode klaster, pendekatan berdasarkan batasan, dan metode meta-heuristic.

2.2.2. Course Timetabling Problem

Course Timetabling memiliki dua kategori yaitu PE-CCT dan CB-CTT. PE-CTT atau singkatan dari *Post Enrolment Course Timetabling* merupakan salah satu permasalahan penjadwalan di bidang pendidikan. Permasalahan PE-CTT yaitu pengalokasian penjadwalan mata kuliah dalam jumlah ruangan dan periode waktu tertentu, di mana konflik antar mata kuliah diatur sesuai dengan pendaftaran mata kuliah yang telah dilakukan oleh mahasiswa. CB-CTT atau singkatan dari *Curriculum Based Course Timetabling* merupakan salah satu permasalahan penjadwalan di bidang pendidikan. Berbeda dengan PE-CTT, CB-CTT merupakan pengalokasian penjadwalan mata kuliah yang disesuaikan dengan aturan kurikulum universitas dalam batasan dalam jumlah ruangan dan periode waktu tertentu.

2.2.3. PE-CTT Pada Dataset ITC 2007

Data yang akan digunakan pada penelitian ini adalah dataset International Timetabling Competition (ITC) 2007. Data dapat ditemukan di http://www.cs.qub.ac.uk/itc2007/. ITC 2007 merupakan sebuah kompetisi optimasi dalam bidang pendidikan. Data diklasifikasikan pada tiga kategori yaitu *Exam Timetabling*, *Post Enrolment Course Timetabling*, dan *Curriculum Based Course Timetabling.* Terdapat 24 *instances* pada dataset ITC-2007. Tabel 2.2 mendeskripsikan mengenai *timeslot, events, rooms, features,* dan *students* pada masing-masing *instances*

Tabel 2.2 Dataset ITC-2007

Problem Instances	Time Slots	Events	Rooms	Features	Students
early1	45	400	10	10	500
early2	45	400	10	10	500
early3	45	200	20	10	1000
early4	45	200	20	10	1000
early5	45	400	20	20	300
early6	45	400	20	20	300
early7	45	200	20	20	500
early8	45	200	20	20	500
late9	45	400	10	20	500
late10	45	400	10	20	500
late11	45	200	10	10	1000
late12	45	200	10	10	1000
late13	45	400	20	10	300
late14	45	400	20	10	300
late15	45	200	10	20	500
late16	45	200	10	20	500
hidden17	45	100	10	10	500
hidden18	45	200	10	10	500
hidden19	45	300	10	10	1000
hidden20	45	400	10	10	1000
hidden21	45	500	20	20	300
hidden22	45	600	20	20	500
hidden23	45	400	20	30	1000
hidden24	45	400	20	30	1000

2.2.4. Model Matematis Dataset ITC 2007

Dari dataset ITC-2007, dapat dirumuskan model matematisnya. Tabel 2.3 mendeskripsikan keterangan notasi yang digunakan pada model matematis ITC-2007

Tabel 2.3 Notasi Model Matematis

Notasi	Keterangan
E	*Event* $(e_1, \dots, e_n)$
S	Mahasiswa $(s_1, \dots, s_n)$
T	Timeslot $(t_1, \dots, t_n)$
R	Ruangan $(r_1, \dots, r_n)$
F	Fitur $(f_1, \dots, f_n)$
K	Kapasitas Ruangan $(k_1, \dots, k_n)$
D	Hari $(1, \dots, 5)$
U	Nilai matriks urutan *event* (-1, 0, 1)
SS	Nilai matriks slot event yang dibolehkan (0, 1)

Berikut uraian mengenai model matematis dataset ITC-2007

a) *Decision Variable*

PE-CTT pada dataset ITC-2007 memiliki sembilan timeslot per hari dalam lima hari, sehingga terdapat 45 timeslot yang harus dialokasikan pada slot waktu dan ruangan. Representasinya sebagai berikut :

$E_{tiri}\{^{1}_{0}$; 1 jika event i pada timeslot t dan ruangan r, 0 jika tidak

$E_{si}\{^{1}_{0}$; 1 jika event i diikuti oleh mahasiswa s, 0 jika tidak

b) *Objective Function*

Setelah variabel keputusan telah ditentukan, selanjutnya adalah menentukan *Objective Function* / fungsi tujuan. Adapun fungsi tujuan dalam dataset ITC-2007 adalah meminimalisasi jumlah penalti yang didapat akibat melanggar *soft constraint*s (SC).

$$z = \min \sum_{i=1}^{3} SC_i \qquad (2.1)$$

c) *Hard Constraint dan Soft Constraint*

Hard Constraint yang harus dipenuhi pada dataset ITC 2007 sebagai berikut :

- tidak ada mahasiswa yang memiliki lebih dari satu event pada waktu yang sama

$$HC_1 = \sum_{r \in R} c_{etr} \, . \, m_{se} \leq 1 \qquad e \in E, s \in S, t \in T \qquad (2.2)$$

dimana,

$$m_{se} = \begin{cases} 1, \text{jika mahasiswa } s \text{ mengikuti } event\ e \\ 0 \end{cases}$$

$$c_{etr} = \begin{cases} 1, \text{jika } event \text{ dijadwalkan ke } timeslot \text{ dan ruangan} \\ 0 \end{cases}$$

- ruangan dapat menampung jumlah mahasiswa yang hadir dan memenuhi semua fitur yang diperlukan

$$HC_2 = k_{er} \, . \, b_{er} \, . \, x_{etr} = x_{etr} \qquad e \in E, r \in R, t \in T \quad (2.3)$$

dimana,

$$k_{er} = \begin{cases} 1, \text{jika ukuran event } e \leq \text{kapasitas ruangan } r \\ 0 \end{cases}$$

$$b_{er} = \begin{cases} 1, \text{jika } \sum_{f \in F} g_{ef} \, . \, h_{rf} = \sum_{f \in F} g_{ef} \\ 0 \end{cases}$$

$$g_{ef} = \begin{cases} 1, \text{jika event } e \text{ membutuhkan features } f \\ 0 \end{cases}$$

$$h_{rf} = \begin{cases} 1, \text{jika ruangan } r \text{ mempunyai features } f \\ 0 \end{cases}$$

- hanya satu event yang dimasukkan ke setiap ruangan di timeslot

$$HC_3 = \sum_{e \in E} c_{etr} \leq 1 \qquad r \in R, t \in T \qquad (2.4)$$

- event dijadwalkan pada timeslot yang diperbolehkan yang sudah ditentukan

$$HC_4 = \sum_{t \in T} c_{etr} \, ss = 1 \qquad e \in E, u \in U \qquad (2.5)$$

- jika ditentukan, event dijadwalkan dalam urutan yang benar dalam seminggu

$$HC_5 = \sum_{t \in T} c_{eu_1} < \sum_{t \in T} c_{eu_2} = 1 \qquad e \in E, u \in U \qquad (2.6)$$

$$HC_5 = \sum_{t \in T} c_{eu_1} > \sum_{t \in T} c_{eu_2} = -1 \qquad e \in E, u \in U \quad (2.7)$$

Penalti diberikan apabila penjadwalan melanggar *Soft Constraint* pada dataset ITC 2007 sebagai berikut :

- seorang siswa memiliki kelas di timeslot terakhir pada hari tersebut

$$SC_1 = \sum_{s \in S} \sum_{t \in \{9,18,\ldots,45\}} y_{st} \quad (2.8)$$

dimana,

$$y_{st} = \begin{cases} 1, \text{jika } c_{etr} \,.\, m_{se} = 1 & r \in R, t \in T \\ 0 \end{cases}$$

- seorang siswa memiliki lebih dari dua kelas secara berurutan

$$SC_2 = \sum_{s \in S} \sum_{d=1}^{5} \sum_{t=(d-1)\times 9+1}^{d\times 9-2} y_{st} \cdot y_{s(t+1)} \cdot y_{s(t+2)} \quad (2.9)$$

- seorang siswa memiliki hanya satu kelas pada satu hari

$$SC_3 = \sum_{s \in S} \sum_{d=1}^{5} y_{sd} \quad (2.10)$$

$$y_{sd} = \begin{cases} 1, \text{jika } \sum_{t=(d-1)\times 9+1}^{d\times 9} y_{st} = 1 & d \in D, s \in S \\ 0 \end{cases}$$

2.2.5. PE-CTT Pada Dataset Socha

Socha merupakan sebuah dataset yang dikembangkan oleh Ben Paechter. Socha memiliki 11 *instances*, yang terdiri dari 5 *instances* berukuran kecil, 5 *instances* berukuran sedang, dan 1 *instances* berukuran besar. Dalam Socha dataset, timeslot yang tersedia sebanyak 45 timeslot dengan rincian 9 timeslot setiap hari dalam 5 hari kerja.

Terdapat tiga *hard constraints* yang harus dipenuhi :

- tidak ada siswa yang mengikuti lebih dari satu mata kuliah di saat yang sama
- ruangan memiliki kapasitas yang dapat menampung jumlah siswa dan memenuhi fitur yang diperlukan mata kuliah tersebut
- hanya satu mata kuliah pada setiap ruangan di setiap timeslot

Sedangkan untuk *soft constraints*-nya, diberikan penalti apabila melanggar (socha min max):

- seorang siswa memiliki kelas di timeslot terakhir pada hari tersebut

- siswa yang memiliki dua mata kuliah berurutan
- siswa yang memiliki hanya satu mata kuliah dalam sehari

Objective Function-nya yaitu meminimalisir angka pelanggaran *soft constraints* dalam sebuah solusi yang layak. Tabel 2.4 berisi mengenai penjelasan jumlah *event*, ruang, *features*, dan mahasiswa pada masing-masing *instance* dataset Socha.

Tabel 2.4 Dataset Socha

Karakteristik	*small*	*medium*	*large*
Jumlah e*vent*	100	400	400
Jumlah ruang	5	10	10
Jumlah f*eatures*	5	5	10
Jumlah mahasiswa	80	200	400

2.2.6. Perbandingan Dataset Socha dan Dataset ITC-2007

Pada penelitian yang akan dilakukan, dataset ITC-2007 dan dataset Socha akan digunakan sebagai dalam penyelesaian masalah *Post Enrolment Course Timetabling*. Format kedua dataset sama, terdapat hal yang berbeda yaitu *hard constraint*. Tabel 2.5 memberikan deskripsi perbedaan batasan dari dataset socha dan dataset ITC-2007.

Tabel 2.5 Perbedaan Hard Constraints Dataset Socha dan ITC2007

		Socha	**ITC-2007**
Hard Constraints	tidak ada siswa yang mengikuti lebih dari satu mata kuliah di saat yang sama	V	V
	ruangan dapat menampung jumlah mahasiswa yang hadir dan memenuhi semua fitur yang diperlukan untuk perkuliahan	V	V
	hanya satu mata kuliah pada setiap ruangan di setiap timeslot	V	V
	Event dijadwalkan pada timeslot yang dibolehkan yang telah ditentukan	X	V
	event dijadwalkan dalam urutan yang benar dalam seminggu	X	V
Soft Constraints	siswa yang memiliki dua mata kuliah berurutan	V	V
	siswa yang memiliki hanya satu mata kuliah dalam sehari	V	V
	siswa memiliki kelas di timeslot terakhir pada hari tersebut	V	V

Dapat dilihat dari tabel 2.5 bahwa hard constraints pada dataset socha dan dataset ITC-2007 memiliki perbedaan yaitu event dijadwalkan pada timeslot yang diperbolehkan dan event dijadwalkan dalam urutan yang benar dalam seminggu. Apabila batasan tidak diimplementasikan maka dapat dikatakan bahwa solusi tidak layak. Perbedaan kompleksitas pada dataset Socha dan dataset ITC-2007 dapat dilihat pada tabel 2.6.

Tabel 2.6 Perbedaan Kompleksitas Dataset Socha dan Dataset ITC-2007

Socha				**Problem Instances**	**ITC-2007**			
Events	**Rooms**	**Features**	**Students**		**Events**	**Rooms**	**Features**	**Students**
100	5	5	80	**1**	100	10	10	500
100	5	5	80	**2**	200	10	10	500
100	5	5	80	**3**	200	10	20	500
100	5	5	80	**4**	200	10	20	500
100	5	5	80	**5**	400	20	10	300
400	10	5	200	**6**	400	20	10	300
400	10	5	200	**7**	400	20	20	300
400	10	5	200	**8**	400	20	20	300
400	10	5	200	**9**	200	20	20	500
400	10	5	200	**10**	200	20	20	500
400	10	10	400	**11**	400	10	10	500
				12	400	10	10	500
				13	400	10	20	500
				14	400	10	20	500
				15	200	20	10	1000
				16	200	20	10	1000
				17	200	10	10	1000
				18	200	10	10	1000
				19	300	10	10	1000
				20	400	10	10	1000
				21	500	20	20	300
				22	600	20	20	500
				23	400	20	30	1000
				24	400	20	30	1000

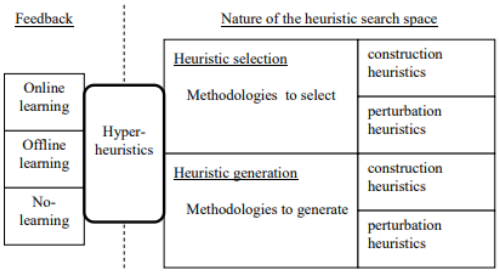
Dapat dilihat dari tabel 2.6 bahwa dataset ITC-2007 lebih kompleks daripada dataset Socha. Perbandingan kompleksitas dapat dilihat dari perbandingan jumlah *events*, ruangan, fitur, dan mahasiswa. Pada dataset Socha, *events* berjumlah 100 hingga 400, sedangkan pada dataset ITC-2007 *events* berjumlah 100 hingga 600.

Ruangan pada dataset Socha berjumlah 5 hingga 10, sedangkan pada dataset ITC-2007 ruangan berjumlah 10 hingga 20. Pada dataset Socha fitur yang dibutuhkan berjumlah 10-20, sedangkan pada dataset ITC-2007 berjumlah 10 hingga 30. Jumlah mahasiswa di ITC-2007 juga lebih banyak yaitu berjumlah 300 hingga 1000, sedangkan pada dataset Socha hanya 80 hingga 400. Hal ini menandakan bahwa dataset ITC-2007 memiliki kompleksitas yang lebih tinggi dibandingkan dataset Socha.

2.2.7. Hyper-Heuristics

Heuristic atau juga biasa disebut dengan istilah *"rule of thumb"* merupakan sebuah metode yang mencari solusi yang baik mendekati optimal namun hasilnya tidak terjamin sebagai solusi yang optimal atau solusi yang layak. *Rule of thumb* maksudnya ialah memberikan petunjuk untuk solusi dari permasalahan. *Metaheuristic* yaitu menerapkan strategi yang dapat memodifikasi heuristik untuk menghasilkan solusi yang lebih optimal dari yang biasanya dihasilkan. Tujuan *hybrid* atau hibridisasi yaitu untuk mengkombinasikan kelebihan dari tipe heuristik yang berbeda. Namun, Burke mengungkapkan bahwa pada bidang penjadwalan diharapkan untuk mengembangkan sistem yang lebih *general* dengan menggunakan metode yang lebih murah dan tetap dapat menyelesaikan masalah. Solusi yang dibutuhkan yaitu solusi yang cukup baik dalam waktu yang cukup cepat. Salah satu solusi yang dapat mengatasi hal tersebut adalah dengan menggunakan pendekatan *Hyper-Heuristic*. *Hyper-heuristics* merupakan metode sederhana yang melakukan pencarian pada *search space heuristic* [2]. Definisi dari *Hyper-heuristic* adalah heuristik yang memilih heuristik atau heuristik yang menghasilkan heuristik. Pengembangan *Hyper-heuristic* memiliki tujuan untuk secara otomatis memecahkan permasalahan menjadi lebih umum. *Hyper-heuristic* merupakan sebuah algoritma pada tingkat yang lebih tinggi yaitu menggunakan heuristik dengan level lebih rendah yang sesuai untuk diterapkan pada masalah yang ada. Sebuah *Hyper-heuristic* berkaitan

dengan eksplorasi dari ruang pencarian heuristik bukan berurusan langsung dengan solusi dari permasalahan tersebut [14].

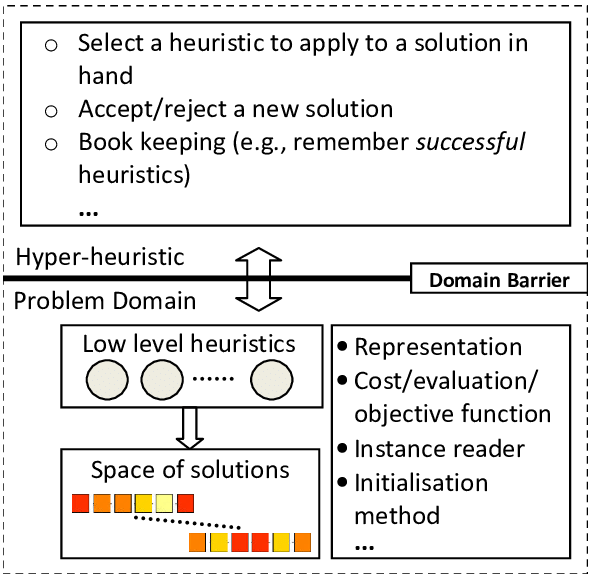


Gambar 2.1 Klasifikasi Pendekatan *Hyper-Heuristics*

Berdasarkan gambar 2.1. pembelajaran *Hyper-heuristics* dapat diklasifikasikan menjadi dua yaitu pembelajaran *online* dan *offline*. Pembelajaran *online* yaitu pembelajaran dilakukan ketika algoritma sedang menyelesaikan permasalahan, sementara *offline learning* yaitu ilmu dikumpulkan membentuk aturan atau program yang dibentuk dari *training instances* dan diharapkan dapat menggeneralisasikan *instances* yang belum terlihat. Terdapat dua metodologi pada pendekatan *Hyper-heuristics* yaitu *heuristic selection* dan *heuristic generation*. *Heuristic Selection* yaitu sebuah metodologi untuk memilih *heuristic* dari *heuristic* yang telah ada. *Heuristic Generation* yaitu sebuah metodologi untuk menghasilkan *heuristic* yang baru dari komponen *heuristic* yang telah ada. Pada metodologi *constructive selection* yaitu dimulai dari solusi kosong kemudian heuristik dipilih dan digunakan untuk menyelesaikan sebuah solusi. Sementara pada metodologi *perturbative selection* pendekatannya adalah memperbaiki solusi kandidat melalui proses pemilihan dan penggunaan heuristik secara otomatis. Pemilihan heuristik tersebut dipilih berdasarkan kesesuaian dengan permasalahan yang dihadapi.

Pada gambar 2.2 menjelaskan mengenai kerangka kerja *Hyper-Heuristics*. Pencarian solusi optimal *Hyper-Heuristic* dilakukan di atas *search space* menggunakan *Low-Level Heuristic*. Hal ini berbeda dengan *metaheuristic* yang bekerja di atas *search space* berupa *solution space*. Hal ini dikarenakan *Hyper-Heuristics* tidak bersinggungan langsung dengan *solution space*, namun dengan

Low-Level Heuristic. Hal tersebut menyebabkan *Hyper-Heuristic* tidak perlu parameter tuning manual untuk setiap problem, karena parameter tuning dilakukan secara otomatis.



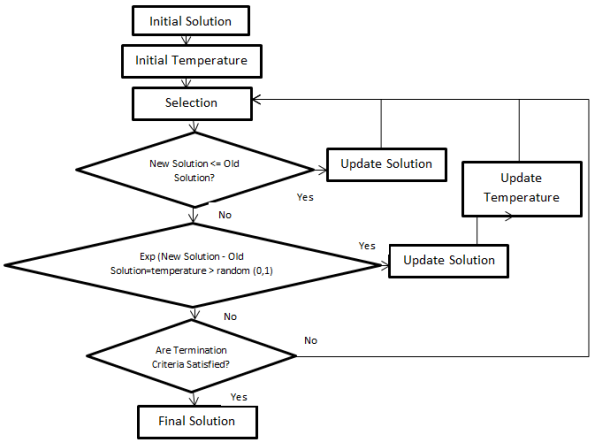
Gambar 2.2 Gambar 2.2 Hyper-Heuristics Framework

2.2.8. Simulated Annealing

Algoritma *Simulated Annealing* merupakan algoritma pencarian lokal yang pertama kali digagas oleh Kirkpatrick pada 1983. Ide awal *Simulated Annealing* adalah untuk menghindari terjebak pada *local optima*, yaitu dengan menerima solusi yang tidak lebih baik apabila tidak melebihi jumlah iterasi tertentu/"suhu" tertentu [2]. Penamaan *Simulated Annealing* diambil dari teori fisika saat proses menguatkan baja. Penguatan baja tersebut dilakukan dengan pemanasan baja hingga mencapai titik didihnya, atom dalam baja akan bergerak bebas. Kemudian baja didinginkan bertahap hingga mencapai titik tentu dengan tujuan energinya berkurang secara perlahan.

Solusi awal pada *Simulated Annealing* didapatkan dengan acak. Langkah berikutnya yaitu, pencarian pada *search space*. Hasil pencarian tersebut dihitung untuk mendapatkan solusi baru. Apabila solusi baru lebih optimal daripada solusi awal, maka solusi akan diterima sebagai solusi sementara. Jika tidak lebih optimal, maka solusi akan melalui fungsi pengontrol. Fungsi pengontrol dalam

Simulated Annealing dinyatakan dengan persamaan Boltzman, yaitu $P = e^{-\frac{c}{t}}$, dimana P adalah Probabilitas Boltzman, e adalah bilangan eksponensial, c adalah perbedaan evaluasi fungsi tujuan antara solusi dengan kandidat solusi, dan t adalah parameter suhu.



Gambar 2.3 *Flowchart Simulated Annealing*

Pseudocode Algoritma Simulated Annealing sebagai berikut :

```
1: procedure Simulated Annealing
2:    current ← initial solution
3:    t ← initial temperature
4:    l ← initial length
5:    repeat
6:     for i = 1 to l do
7:       candidate ϵ N(current)
8:        if f(candidate) <= f(current) then
9:           current ← candidate
10:        else if
11:         exp(f(current) – f(candidate)=t) > random[0,1)then
12:          current ← candidate
13:        end if
14:    end for
15:    update l and t
16:   until stop condition
17: end procedure
```

2.2.9. Tabu Search

Tabu Search merupakan sebuah metode metaheuristik yang dikembangkan oleh Glover pada 1986 [2]. Prinsip dasar dari *Tabu Search* yaitu menerima solusi yang tidak lebih baik, namun tidak dapat kembali ke solusi yang sudah pernah

ditemukan. Solusi yang sudah pernah ditemukan tersebut disimpan dalam memori yang bernama *tabu list*. *Tabu list* menyimpan solusi yang pernah ditemui dan tidak lebih baik dari solusi awal, agar tidak berulang pada area solusi yang sama.

Dalam implementasinya, setiap iterasi atau setiap perpindahan timeslot akan memeriksa *tabu list*. Apabila solusi tersebut terdapat pada *tabu list*, maka solusi tersebut tidak akan diambil sebagai solusi baru sehingga mencegah solusi berulang [15]. Namun, jika solusi tabu dapat menghasilkan solusi yang lebih baik dari solusi sebelumnya, status dalam tabu list dapat dibatalkan. Pengelolaan tabu list menggunakan aturan FIFO (*First-in First-Out*), yang artinya apabila terdapat solusi tabu baru, maka solusi tabu paling awal dihapus.

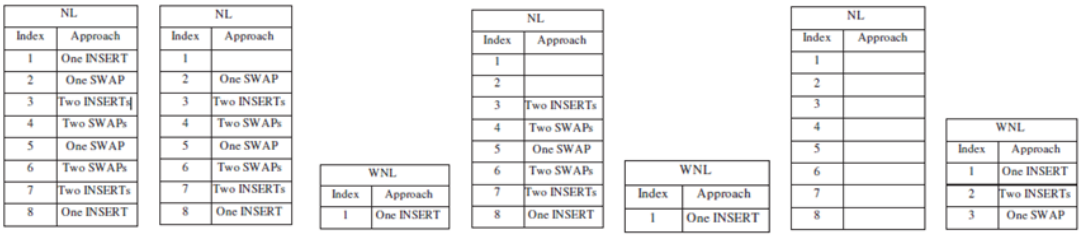
```
1: procedure Tabu Search
2:   Tabu list T
3:   current ← initial solution
4:   best ← current
5:    repeat
6:      current ← arg min_{x∈N(current)} f(x), x: non-tabu
7:      if current < best then
8:         best ← current
9:      end if
10:        record the recent move in T
           delete the oldest entry if necessary
11:   until stop condition
12: end procedure
```

2.2.10. Self-Adaptive

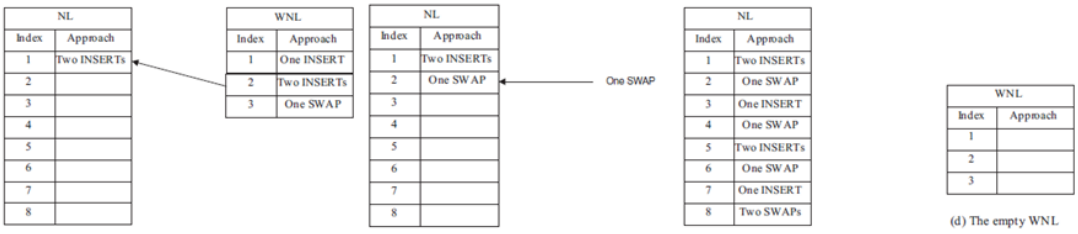
Mekanisme *self-Adaptive* memiliki struktur yang sederhana dan kinerja yang dinamis sehingga dapat meningkatkan kinerja *search algorithm*. Xiangtao Li menerapkan self-adaptive untuk menghasilkan solusi yang lebih baik dengan adanya pengetahuan mengenai nilai-nilai sebelumnya yang sukses [16]. *Self-adaptive* dapat meningkatkan tingkat konvergensi ke nilai optimal pada keseluruhan proses optimisasi [9]. Liu et al menggunakan strategi *self-adaptive* pada algoritma *Artificial Bee Colony* agar dapat mengarahkan setiap individu lebah ke nilai yang optimal.

Self-Adaptive juga dikombinasikan dengan *Artificial Bee Colony* diterapkan oleh Quan-ke Pan dalam menyelesaikan permasalahan *lot-streaming flow shop* [8]. Konsep dasar dari *Self-Adaptive* adalah dengan memiliki

Neighboring List (NL) yang didapat dari *Heuristic* yang ditetapkan. Kemudian apabila solusi masing-masing List lebih baik maka akan diterima di Winning List (WL). Apabila NL kosong, maka WL akan dipindah kembali ke NL, ditambah dengan random tindakan *Heuristic*, begitu seterusnya.



Gambar 2.4 Ilustrasi Strategi Self-Adaptive NL dan WNL [8]



Gambar 2.5 Lanjutan Ilustrasi Strategi Self-Adaptive [8]

Pada gambar 2.4 dan 2.5 menggambarkan ilustrasi strategi Self-Adaptive. Langkah pembentukan NL dan WNL digambarkan pada gambar 2.4. Pada gambar 2.5 digambarkan bahwa dari beberapa NL di awal, selanjutnya yang digunakan adalah WNL yang telah terbentuk.

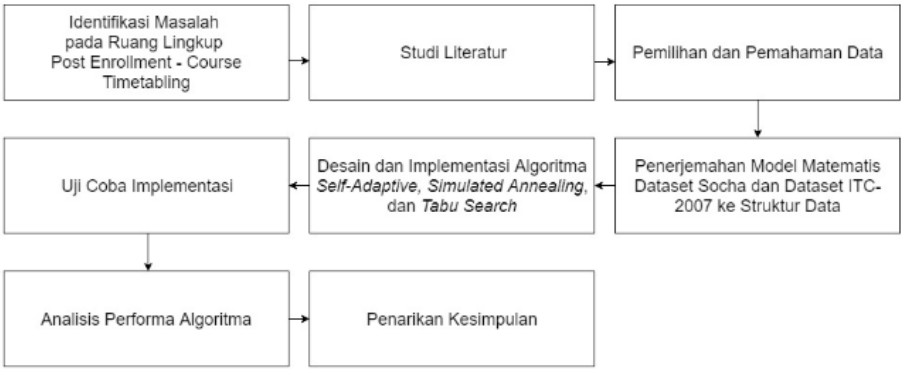
```
1: procedure Self-Adaptive
2:   Neighboring list NL
3:   Winning Neighboring list WNL
4:   totalNL = 8
5:   totalWNL = 0
5:   current <- initial solution
6:   repeat
7:    for i = 1 to totalNL do
8:      candidate ϵ N(current)
9:      if f(candidate) <= f(current) then
10:        current <- candidate
11:        update WNL
12:        totalWNL++
13:     end if
14:   end for
15:   for j = 1 to totalWNL do
```

```
16:      NL <- random[WNL]
17:      WNL <- null
18:    end for
19:  until stop condition
20: end procedure
```

Halaman ini sengaja dikosongkan

BAB 3
METODOLOGI PENELITIAN

Bab ini akan menjelaskan langkah-langkah yang diperlukan dalam proses penelitian. Terdapat dua sub-bab yang akan dijelaskan yaitu tahapan penelitian dan rencana penelitian. Tahapan penelitian ditampilkan mulai tahapan identifikasi masalah hingga penarikan kesimpulan. Rencana penelitian ditampilkan dalam bentuk tabel waktu yang berisi informasi perkiraan waktu (dalam minggu/bulan) untuk pengerjaan tesis pada gambar 3.1.



Gambar 3.1 Tahapan Penelitian

3.1. Identifikasi Masalah

Tahap pertama yaitu tahap pengidentifikasian masalah, topik permasalahan penjadwalan pada bidang pendidikan ditetapkan menjadi topik utama yang akan dibahas. Terdapat tiga jenis penjadwalan pada bidang pendidikan, yaitu :

a. *Examination Timetabling*
b. *Post-Enrolment Course Timetabling*
c. *Curriculum Based Course Timetabling*

Terdapat penelitian mengenai *Post Enrolment Course Timetabling* yang telah dikembangkan oleh Ahsanul. Penelitian tersebut masih dapat dikembangkan agar dapat menangani dataset lain yang ditetapkan dan mampu memberikan hasil yang

optimal. Sehingga pada penelitian ini ditetapkan untuk fokus pada *Post Enrolment Course Timetabling* dan mengembangkan penelitian oleh Ahsanul.

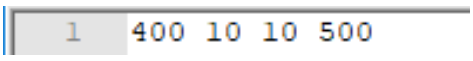
3.2. Studi literatur

Pada tahap studi literatur, penelitian-penelitian terkait Penjadwalan, *University Timetabling Problem*, dan *Post Enrolment Course Timetabling* dan metode heuristik yang telah dilakukan sebelumnya dikaji. Pengkajian terhadap penelitian terdahulu tersebut dapat memberikan referensi terkait pengembangan metode dan analisis kelebihan dan kekurangan metode yang telah digunakan sebelumnya. Analisis gap penelitian dan potensi penelitian selanjutnya didapatkan dari hasil pengkajian.

3.3. Pemilihan dan Pemahaman Data

Setelah tahap studi literatur dilakukan, selanjutnya adalah tahap pemilihan dan pemahaman data. Pada penelitian sebelumnya dataset yang digunakan adalah dataset Socha. Dataset lainnya yang dipilih dalam penelitian ini untuk mengembangkan dan menggeneralisasikan aplikasi tersebut adalah dataset ITC 2007. Dataset ITC 2007 dipilih karena memiliki tingkat kompleksitas yang lebih banyak daripada dataset Socha dan merupakan salah satu dataset yang populer dan banyak dibahas dalam bidang penjadwalan universitas.

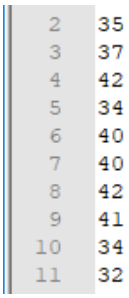
Kedua dataset tersebut digunakan dengan berfokus pada *domain Post Enrolment Course Timetabling.* Format penulisan dataset Socha dan ITC 2007 pada domain tersebut terdapat kemiripan, namun dataset ITC-2007 memiliki masukan yang lebih kompleks. Baris pertama yaitu jumlah events; jumlah ruangan; jumlah features; jumlah mahasiswa seperti misalnya yang terlihat pada gambar 3.2.



Gambar 3.2 Contoh Baris Pertama Format Dataset

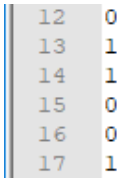
Dapat dilihat pada gambar 3.2, terdapat 400 events, 10 ruangan, 10 features, dan 500 mahasiswa. Kemudian baris selanjutnya menjelaskan kapasitas dari masing-masing ruangan tersebut. Banyaknya baris setelah baris pertama tersebut

menyesuaikan jumlah ruangan yang telah didefinisikan. Apabila ditetapkan bahwa terdapat 10 ruangan maka baris ke-dua hingga baris ke-sebelas mendeskripsikan kapasitas dari ruangan tersebut. Contoh data kapasitas per ruangan dapat dilihat pada gambar 3.3.



Gambar 3.3 Baris Kapasitas Masing-Masing Ruangan

Gambar 3.3 mendeskripsikan mengenai kapasitas masing-masing ruangan. Ruangan pertama mampu menampung 35 mahasiswa, ruangan kedua mampu menampung 37 mahasiswa, begitu seterusnya. Hingga yang terakhir, ruangan kesepuluh mampu menampung 32 mahasiswa. Baris selanjutnya mendeskripsikan mengenai mahasiswa per mata kuliah. Apabila mahasiswa mengambil mata kuliah, maka nilainya 1 jika tidak 0. Data diurutkan dari per mahasiswa kemudian per mata kuliah. Misal terdapat 2 siswa dengan 3 event seperti yang digambarkan pada gambar 3.4.



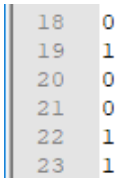
Gambar 3.4 Mahasiswa dan Mata Kuliah

Gambar 3.4 menjelaskan bahwa mahasiswa1 mengikuti mata kuliah ke-2 saja. Sedangkan mahasiswa2 mengikuti mata kuliah ke-1 dan ke-3. Cara membacanya adalah per mahasiswa dan per mata kuliah. Dari format tersebut, dapat diterjemahkan ke matriks mahasiswa dan mata kuliah seperti pada tabel 3.1.

Tabel 3.1 Matriks Mahasiswa dan Mata Kuliah

	Mata Kuliah 1	**Mata Kuliah 2**	**Mata Kuliah 3**
Mahasiswa 1	0	1	0
Mahasiswa 2	1	0	1

Baris selanjutnya memberikan data ruangan dan *features*. Apabila pada ruangan memenuhi features maka nilainya 1, 0 jika tidak. Misal terdapat 2 ruangan dan 3 *features* seperti yang terdapat pada gambar 3.5.



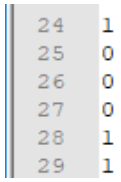
Gambar 3.5 Ruangan dan *Features*

Gambar 3.5 menjelaskan bahwa ruangan1 memenuhi *features* ke-3 saja. Sedangkan ruangan2 memenuhi *features* ke-1 dan ke-3. *Features* ke-2 tidak terdapat di ruangan1 maupun ruangan2. Cara membacanya adalah per ruangan dan per *features*. Dari format tersebut, dapat diterjemahkan ke matriks ruangan dan *features* seperti pada tabel 3.2.

Tabel 3.2 Matriks Ruangan dan *Features*

	***Features* 1**	***Features* 2**	***Features* 3**
Ruangan 1	0	0	1
Ruangan 2	1	0	1

Baris selanjutnya memberikan data mata kuliah dan *features* yang dibutuhkan. Apabila pada mata kuliah memerlukan features maka nilainya 1, 0 jika tidak. Misal terdapat 2 mata kuliah dan 3 *features* seperti yang terdapat pada gambar 3.6.



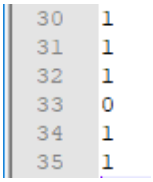
Gambar 3.6 Mata Kuliah dan *Features*

Gambar 3.6 menjelaskan bahwa mata kuliah1 memerlukan *features* ke-1 dan ke-3. Sedangkan mata kuliah2 memerlukan *features* ke-3 saja. Mata kuliah2 tidak memerlukan *features* apapun. Cara membacanya adalah per mata kuliah dan per *features*. Dari format tersebut, dapat diterjemahkan ke matriks mata kuliah dan *features* seperti pada tabel 3.3.

Tabel 3.3 Matriks mata Kuliah dan Features

	***Features* 1**	***Features* 2**	***Features* 3**
Mata Kuliah 1	1	0	1
Mata Kuliah 2	0	0	1

Baris selanjutnya memberikan data mata kuliah dan *timeslot*. Apabila pada mata kuliah diperbolehkan diletakkan pada *timeslot* maka nilainya 1, 0 jika tidak. Misal terdapat 2 mata kuliah dan 3 *timeslot* seperti yang terdapat pada gambar 3.7.



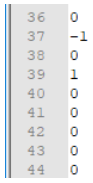
Gambar 3.7 Mata Kuliah dan *timeslot*

Gambar 3.7 menunjukkan bahwa mata kuliah1 diperbolehkan diletakkan pada *timeslot* ke-1, ke-2, dan ke-3. Sedangkan mata kuliah2 diperbolehkan diletakkan pada *timeslot* ke-2 dan ke-3. Cara membacanya adalah per mata kuliah dan per *timeslot*. Dari format tersebut, dapat diterjemahkan ke matriks mata kuliah dan *timeslot* seperti pada tabel 3.4.

Tabel 3.4 Matriks Mata Kuliah dan *Timeslot*

	***Timeslot* 1**	***Timeslot* 2**	***Timeslot* 3**
Mata Kuliah 1	1	1	1
Mata Kuliah 2	0	1	1

Baris selanjutnya memberikan urutan penempatan mata kuliah, apakah sebelum/sesudah/tidak ada preferensi terhadap mata kuliah lainnya. mata kuliah pertama harus dijadwalkan sebelum mata kuliah ke-dua maka nilainya 1. Nilainya -1 apabila mata kuliah pertama harus dijadwalkan setelah mata kuliah ke-dua. Apabila tidak ada preferensi pada mata kuliah pertama dengan mata kuliah ke-dua maka nilainya 0. Misal terdapat 3 mata kuliah seperti yang terdapat pada gambar 3.8.



Gambar 3.8 Urutan Mata Kuliah

Gambar 3.8 menunjukkan bahwa mata kuliah1 harus diletakkan sebelum mata kuliah ke-dua. Sedangkan mata kuliah2 harus diletakkan setelah mata kuliah1. Mata kuliah3 tidak memiliki preferensi apapun terhadap mata kuliah1 dan mata kuliah2. Cara membacanya adalah per mata kuliah. Dari format tersebut, dapat diterjemahkan ke matriks urutan mata kuliah seperti pada tabel 3.5.

Tabel 3.5 Matriks Urutan Mata Kuliah

	Mata Kuliah 1	**Mata Kuliah 2**	**Mata Kuliah 3**
Mata Kuliah 1	0	-1	0
Mata Kuliah 2	1	0	0
Mata Kuliah 3	0	0	0

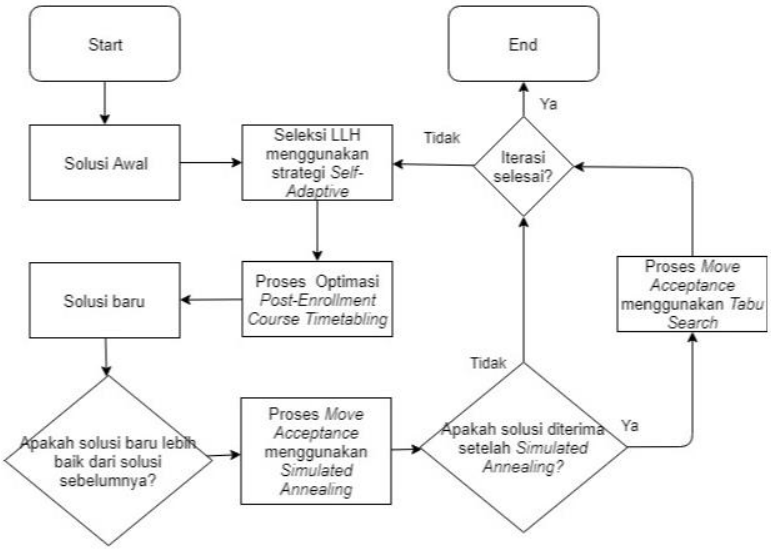
3.4. Penerjemahan Model Matematis ke Struktur Data

Model matematis dataset Socha dan dataset ITC 2007 yang telah didapatkan, diterjemahkan ke dalam struktur data bahasa pemrograman. Model matematis yang akan diterjemahkan adalah domain *Post Enrolment Course Timetabling* dari kedua dataset tersebut. Model matematis pada permasalahan riset operasi terdiri dari dua jenis yaitu fungsi tujuan dan batasan yang telah dibahas pada bab 2 Tinjauan Pustaka.

3.5.Desain dan Implementasi Algoritma

Pada tahap desain dan implementasi algoritma, algoritma yang digunakan yaitu *Self-Adaptive, Tabu Search,* dan *Simulated Annealing* dikombinasikan untuk menyelesaikan permasalahan *Post Enrolment Course Timetabling* pada dataset Socha dan ITC 2007. Flowchart desain algoritma digambarkan pada gambar 3.9.

Solusi awal (*initial solution*) akan terbentuk setelah program membaca file data yang diproses menggunakan *low level heuristic*. Solusi awal tersebut selanjutnya akan diiterasikan dalam proses optimasi. Setelah solusi awal didapatkan, tahap selanjutnya adalah tahap optimasi dengan pemilihan *low level heuristic*. Seperti yang telah dijelaskan bahwa *low level heuristic* yang akan digunakan dipilih dengan strategi *Self-Adaptive*. Kemudian *Move Acceptance*-nya menggunakan *Simulated Annealing* dan *Tabu Search*.



Gambar 3.9 Flowchart metode Hyper-Heuristics yang digunakan

Self-Adaptive akan diimplementasikan pada *local search*, yaitu menggunakan empat pendekatan *neighboring* : *move* satu slot, *swap* satu slot, *move* dua slot, dan *swap* dua slot. Kemudian acceptance criteria dari *Simulated Annealing* akan digunakan. *Simulated Annealing* akan menerima solusi yang lebih baik dan juga menerima solusi yang tidak lebih baik dengan penghitungan proses *annealing*. Solusi yang lolos pada proses *annealing* akan diproses dengan *Tabu Search* untuk memeriksa apakah terdapat solusi yang terdapat dalam daftar pengecualian (*Tabu List*). Solusi yang tidak terdapat dalam *tabu list* akan diterima sebagai solusi baru dengan memasukkan struktur solusi ke dalam *tabu list*, sehingga struktur lingkungan tersebut tidak dapat digunakan kembali pada iterasi-iterasi berikutnya. Solusi yang dihasilkan pada satu iterasi tersebut dapat disebut dengan istilah Daftar Pemenang (*Winning List*). Iterasi berikutnya, Daftar Pemenang ditambah dengan beberapa random pendekatan *neighboring* yang akan dioptimasi. Suhu dalam proses *Simulated Annealing* akan diturunkan setiap iterasi terjadi.

Penggunaan algoritma *Simulated Annealing*, *Tabu Search*, dan *Self-Adaptive* memungkinkan diversifikasi spesifik, yaitu dapat menerima solusi yang tidak lebih baik sehingga solusi akhir tidak terjebak dalam *local optima solution*

dan solusi dapat memperbaiki diri sendiri dengan strategi *Self-Adaptive*. Penggabungan ketiga algoritma tersebut juga dapat memberikan solusi yang lebih variatif.

Tahapan dalam pembuatan solusi awal yaitu dengan metode konstruktif. Metode konstruktif merupakan metode yang mengembangkan sebuah solusi dari nol. Algoritma yang digunakan dalam pembuatan solusi awal adalah *Greedy*. Tahapan pembuatan solusi awal adalah sebagai berikut :

1) Mata kuliah yang terdapat pada daftar mata kuliah satu persatu diletakkan pada slot yang tersedia sehingga semua mata kuliah memiliki slot
2) Pada tahap tersebut yang diperhatikan hanya hard constraint, perhitungan penalti atau *soft constraint* diabaikan
3) Pemeriksaan mata kuliah yang memiliki konflik
4) Apabila terdapat mata kuliah yang memiliki konflik, maka timeslot dan ruangan yang masih tersedia dicari
5) Langkah ke tiga dan ke empat diulang hingga tidak ada konflik dan daftar mata kuliah telah memiliki slot dan ruangan. hal ini menunjukkan bahwa solusi awal telah terbentuk

3.6.Uji Coba Implementasi

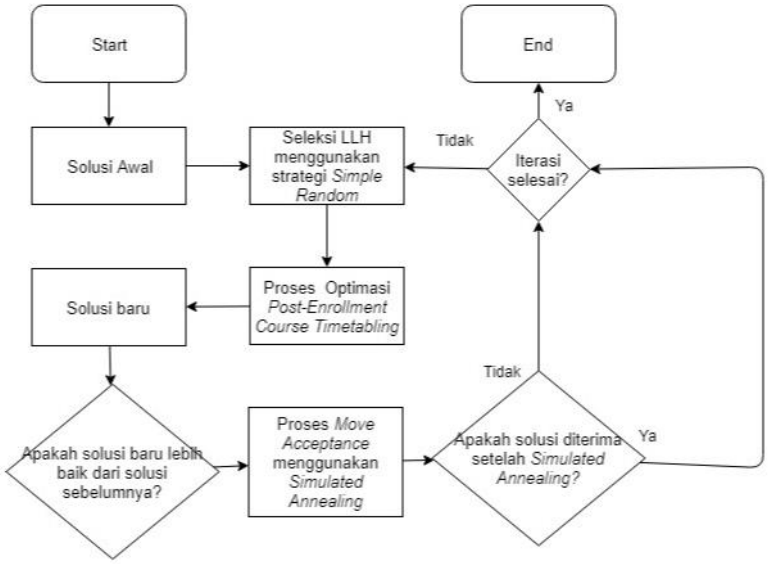
Setelah implementasi selesai dilakukan maka tahap selanjutnya adalah tahap uji coba. Tahapan ini dilaksanakan untuk memastikan bahwa aplikasi dapat dijalankan sesuai fungsinya dan menguji performa algoritma yang digunakan. Tahapan uji coba terdiri dari :

a. Menentukan lingkungan uji coba, yaitu peralatan yang digunakan saat proses uji coba. Spesifikasi komputer yang digunakan harus dideskripsikan.
b. Menentukan dan mengatur kombinasi-kombinasi algoritma yang dapat digunakan dari ketiga algoritma *Self-Adaptive, Simulated Annealing,* dan *Tabu Search* untuk mengetahui kinerja algoritma.

c. Menentukan dan mengatur parameter seperti batasan waktu yang digunakan dalam menjalankan algoritma untuk menyelesaikan permasalahan.

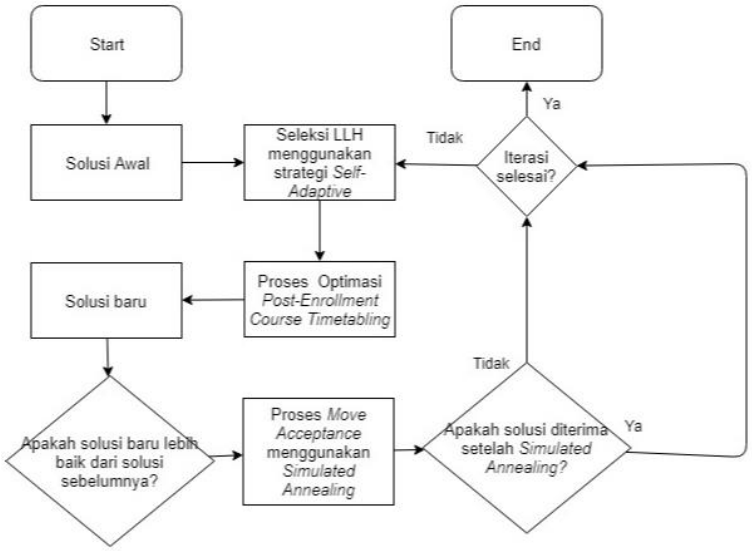
Skenario yang akan digunakan dalam menguji coba implementasi yaitu :

1. Pemilihan LLH (*Low Level Heuristics*) dengan strategi *Simple Random* dengan *Move Acceptance Simulated Annealing.* Pada skenario ini yang membedakan dari skenario biasa adalah digunakannya *simple random* dalam pemilihan *LLH* dan hanya menggunakan *Move Acceptance Simulated Annealing* tanpa *Tabu Search*. LLH yang digunakan dalam proses optimasi akan dipilih dengan diacak secara random. Flowchart skenario yang digunakan digambarkan pada gambar 3.10.



Gambar 3.10 Skenario Uji Coba 1

2. Pemilihan LLH dengan strategi *Self-Adaptive* dengan *Move Acceptance Simulated Annealing.* Pada skenario ini yang membedakan dari skenario biasa adalah digunakannya *Move Acceptance Simulated Annealing* tanpa *Tabu Search*. LLH yang digunakan dalam proses optimasi akan dipilih dengan strategi *Self-Adaptive*. Flowchart skenario yang digunakan digambarkan pada gambar 3.11.



Gambar 3.11 Skenario Uji Coba 2

3. Pemilihan LLH dengan strategi *Self-Adaptive* dengan *Move Acceptance Tabu Search dan Simulated Annealing. S*kenario ini merupakan skenario biasa yaitu yang telah dijelaskan dan digambarkan pada sub bab 3.5. Desain dan Implementasi Algoritma.

3.7.Analisis Performa Algoritma

Setelah uji coba implementasi selesai dilaksanakan, analisis performa algoritma adalah tahap selanjutnya. Pada tahap ini, peneliti akan menganalisa performa algoritma yang diusulkan dan hasil solusi yang didapatkan. Perbandingan hasil solusi dapat dilakukan dengan *problem domain* dan dataset yang sama.

BAB 4
IMPLEMENTASI

Pada bab implementasi dibahas mengenai implementasi algoritma Self-Adaptive-Tabu-Simulated Annealing Hyper-Heuristic terhadap Socha dan ITC-2007 dataset dengan identifikasi permasalahan seperti dijelaskan dalam Bab I. Subbab berikut menjelaskan implementasi algoritma dalam penelitian ini.

4.1. Pembentukan Solusi Awal

Pembentukan solusi awal merupakan proses awal implementasi. Proses pembentukan solusi awal di antaranya yaitu membaca file masukan, membuat matriks diantaranya *conflict matrix*, matriks *conflict course*, matriks *suitable rooms*, matriks *event feature*, dan matriks *student event*. Kelima matriks tersebut dibutuhkan pada kedua domain dataset Socha dan ITC-2007. Tambahan beberapa matriks seperti matriks *suitable slot*, matriks *suitable order*, matriks *after slot* dan matriks *before slot* pada dataset ITC-2007 dikarenakan adanya tambahan *Hard Constraints* yang harus dipenuhi.

4.1.1. Pembentukan Matriks *Suitable Slot*

Matriks Suitable Slot dibentuk untuk memudahkan pengecekan terhadap Hard Constraints ke-empat, yaitu *event* hanya ditugaskan ke slot waktu yang telah ditentukan sebelumnya. Penentuan *event* dengan slot waktu sudah ditetapkan pada dataset ITC-2007. Matriks dua dimensi dengan kolom pertama adalah event dan kolom kedua adalah slot waktu. Data berisi 0, apabila event tersebut tidak dibolehkan diletakkan pada slot waktu tersebut. Data berisi 1, apabila event tersebut dibolehkan diletakkan pada slot waktu tersebut.

```
int[][] suitableSlot() {
   SuitableSlot = new int[event][timeslot];
   for (int i = 0; i < event; i++) {
       for (int j = 0; j < timeslot; j++) {
           SuitableSlot[i][j]=eventSlot[i][j];
             System.out.println("Event ke "+i+", timeslot ke-"+j+" : "+eventSlot[i][j]);
       }
   }
   return SuitableSlot;
}
```

Gambar 4.1 Pembentukan Matriks Suitable Slot

4.1.2. Pembentukan Matriks Suitable Order

Matriks Suitable Order dibentuk untuk memudahkan pengecekan terhadap Hard Constraints ke-lima, yaitu jika ditentukan, event dijadwalkan dalam urutan yang benar dalam seminggu. Matriks dua dimensi dengan kolom pertama adalah event pertama yang akan dibandingkan dan kolom kedua adalah event kedua yang akan dibandingkan. Data dari matriks tersebut akan berisi 0 apabila tidak ada penentuan urutan. Data bernilai 1 apabila event pertama dijadwalkan setelah event kedua, dan bernilai -1 apabila event kedua dijadwalkan sebelum event pertama.

```
int[][] suitableOrder() {
   SuitableOrder = new int[event][event];
   for (int i = 0; i < event; i++) {
       for (int j = 0; j < event; j++) {
           SuitableOrder[i][j]=eventOrder[i][j];
             System.out.println("Event ke "+i+", timeslot ke-"+j+" : "+eventOrder[i][j]);
       }
   }
   return SuitableOrder;
}
```

Gambar 4.2 Pembentukan Matriks Suitable Order

4.1.3. Pembentukan Matriks Before Slot

Matriks Before Slot dibentuk untuk memudahkan pengecekan panjang / jumlah total *event* yang harus diletakkan sebelum *Event* i diletakkan. Data dari matriks before slot berisi *event-event* yang harus dilaksanakan sebelum *event* i diadakan.

```
int[][] beforeSlot(){
    beforeSlot = new int[event][];
    for (int i = 0; i < event; i++) {
        ArrayList<Integer> temp = new ArrayList<Integer>();
        for (int j = 0; j < SuitableOrder[i].length; j++){
            int conflict = SuitableOrder[i][j];
              System.out.println("int conflict " + conflict);
              System.out.println("i "+i+"j " + j); //j adalah event ke-
            if (conflict < 0 && i != j) {
                temp.add(j);
                  System.out.println("temp " + temp);
            }
        }
        int[] arrayTemp = new int[temp.size()];
        for (int j = 0; j < temp.size(); j++) {
            arrayTemp[j] = temp.get(j);
              System.out.println("arrayTemp "+arrayTemp[j]);
        }
        beforeSlot[i] = arrayTemp;
    }

    return beforeSlot;
}
```

Gambar 4.3 Pembentukan Matriks Before Slot

4.1.4. Pembentukan Matriks After Slot

Matriks After Slot dibentuk untuk memudahkan pengecekan panjang / jumlah total *event* yang diletakkan setelah *Event* i diletakkan. Data dari matriks after slot berisi, *event-event* yang dilaksanakan setelah event i diadakan.

```
int[][] afterSlot(){
    afterSlot = new int[event][];
    for (int i = 0; i < event; i++) {
        ArrayList<Integer> temp = new ArrayList<Integer>();
        for (int j = 0; j < SuitableOrder[i].length; j++){
            int conflict = SuitableOrder[i][j];
              System.out.println("int conflict " + conflict);

            if (conflict > 0 && i != j) {
                temp.add(j);
                  System.out.println("temp " + temp);
                  System.out.println("i "+i+"j " + j); //j adalah event ke-
            }
        }
        int[] arrayTemp = new int[temp.size()];
        for (int j = 0; j < temp.size(); j++) {
            arrayTemp[j] = temp.get(j);
              System.out.println("arrayTemp "+arrayTemp[j]);
        }
        afterSlot[i] = arrayTemp;
    }
    return afterSlot;
}
```

Gambar 4.4 Pembentukan Matriks After Slot

4.1.5. Pengurutan Event untuk Inisiasi Awal

Event yang memiliki penetapan urutan slot diprioritaskan dalam pembentukan solusi awal. Event yang memiliki urutan slot paling banyak didahulukan agar lebih memudahkan dalam penjadwalan solusi awal.

```
for (int i = 0; i < conflictcourse.length; i++) {
      hitunganTimeslot[i]=timeslot;
    Collections.sort(course, new courseChained(
        new courseSortingOrderSlot(),
        new courseSortingStudent(),
        new courseSortingRooms(),
        new courseSortingConflict(),
        new courseSortingRandom()
        ));
```

Gambar 4.5 Pengurutan Event untuk Inisiasi Awal

Event yang memiliki urutan ditempatkan dalam sebuah arrayList. Pada arrayList tersebut, event yang diletakkan paling depan dicek dahulu apakah memiliki syarat event sebelumnya. Begitu halnya dengan event yang diletakkan paling belakang dicek apakah memiliki syarat event sesudah event tersebut. Hal ini sesuai dengan pseudocode Create Array Order Slot.

Pseudocode 4.1 Create Array Order Slot

```
1: procedure Create Array Order Slot
2:    index ← no event / course
3:    beforeslot ← matrix contains event that must be scheduled
       before index
4:    afterslot ← matrix contains event that must be scheduled
       after index
5:    if index has beforeslot or afterslot then
6:      add index to array Order Slot
7:        if index has beforeslot then
8:         for j = 1 to beforeslot size do
9:             add beforeslot[index][j] to array Order Slot,
            before index in array Order Slot
10:        end for
11:       end if
12:       if index has afterslot then
13:         for j = 1 to afterslot size do
14:          add afterslot[index][j] to array Order Slot, after
              index in array Order Slot
15:         end for
16:       end if
17:      //array Order Slot = {event beforeslot(1..n), index,
           event afterslot(1..n)}
18:   until stop condition
19: end procedure
```

```
if (beforeslot[index].length>0 || afterslot[index].length>0){
    reqEvent = new ArrayList<>();
    reqEvent.add(index);
    if (beforeslot[index].length>0){
        for (int j = 0; j < beforeslot[index].length; j++) {
        reqEvent.add(0, beforeslot[index][j]);
          System.out.println("req eventnya first "+reqEvent);
        }
    }
    if (afterslot[index].length>0){
        for (int j = afterslot[index].length-1; j < afterslot[index].length; j++) {
            reqEvent.add(reqEvent.size(), afterslot[index ][j]);
        }
    }
```

Gambar 4.6 Pengecekan Urutan Event

4.1.6. LLH Inisiasi Awal

Terdapat tiga metode dalam pembentukan inisiasi awal yaitu mencari timeslot dengan urut awal, acak, dan urut akhir. Pengurutan event dari awal berdasarkan slot minimal dan maksimal sehingga dapat tidak bertabrakan dengan event sebelumnya atau sesudahnya. Berikut pseudocode urut awal

Pseudocode 4.2 First Sort Initial Solution

```
1: procedure First Sort Initial Solution
2:    totaltimeslot ← 45
3:    timeslot ← matrix event and slot
4:    index ← no event
5:   indexconflict ← event conflict with index, can't be in the
     same slot
6:     suitableSlot ← matrix that contains event with
     suitableslot, event can be scheduled on timeslot if 1
7:   timeslotrooms ← matrix slot and room
8:   suitabeRoom ← matrix event and room, room is suitable for
     event if 1
9:   room ← matrix room and index
10:  repeat
11:    for i = 1 to totaltimeslot do
12:      if suitableSlot = 1 then //procedure searchTS
13:       if index timeslot != indexconflict timeslot then
          //procedure searchTS
14:              timeslot[index] = i //procedure searchTS
15:          for j = 1 to timeslotrooms[i].length do
16:             if timeslotRooms slot i room j isEmpty then
17:               if suitableRoom event index room j isSuitable
                    then
18:                 place room[index]=j
19:               end if
20:             else if timeslotRooms slot i room j isNotEmpty
21:               return false
22:             end if
23:          end for
24:       end if
```

```
25:      else if suitableSlot = 0 then
26:        return false
27:      end if
28:    end for
29: end procedure
```

```
void firstSortInitialTS (int index, int currentSlot, int[][] conflictcourse, int[][] timeslotrooms, int[][] suitablerod
    int timeslot, int[][] suitableslot, int[][] afterslot, int[][] beforeslot){
outerloop:
for (int i =0; i < timeslot; i++){
    if (searchTS(index, (currentSlot+1), conflictcourse, suitableslot, afterslot, beforeslot)) { //kalau searchTS
        for (int k = 0; k < timeslotrooms[currentSlot].length; k++) {
            //cek contraint mengenai timeslot dan ruangan
            if (cekTimeslotRooms(index, timeslotrooms, suitablerooms, currentSlot, k)){
                //apabila lolos cek constrain, taruh timeslot dan ruangan
                placeTimeslotRooms(index, timeslotrooms, currentSlot, k);
                 System.out.println("Event "+ index +" ditaruh courseTimeslot " + (j+1));
                break outerloop;
            }
        }
    }
    currentSlot++;
}
}
```

Gambar 4.7 First Sort Initial TS

Pengurutan event dengan acak berdasarkan range arrayOrderSlot yang sudah ditetapkan. Berikut pseudocode acak

Pseudocode 4.3 Random Sort Initial TS

```
1: procedure Random Sort Initial Solution
2:   index ← no event
3:   indexconflict ← event conflict with index, can't be in the
     same slot
4:   timeslot ← matrix event and slot
5:   low ← event no in arrayorderSlot*(timeslot/array
     arrayorderSlot size)
6:   high ← (event no in arrayorderSlot+1)*(timeslot/array
     arrayorderSlot size)
7:   random ← rand(high-low) + low
8:     suitableSlot ← matrix that contains event with
     suitableslot, event can be scheduled on timeslot if 1
9:   timeslotrooms ← matrix slot and room
10:   suitabeRoom ← matrix event and room, room is suitable for
     event if 1
11:   room ← matrix room and index
12:   repeat
13:    for i = 1 to 100 do
14:      if suitableSlot = 1 then //procedure searchTS
15:       if index timeslot!= indexconflict timeslot then
          //procedure searchTS
16:          timeslot[index] = random //procedure searchTS
17:          for j = 1 to timeslotrooms[random].length do
18:            if timeslotRooms slot random room j isEmpty then
19:               if suitableRoom event index room j isSuitable
                  then
```

```
20:                    place room[random]=j
21:                  end if
22:                    else if timeslotRooms slot random room j
                     isNotEmpty then
23:                  return false
24:               end if
25:             end for
26:           end if
27:        else if suitableSlot = 0 then
28:          return false
29:        end if
30:        update random
31:      end for
32:    until stop condition
33: end procedure
```

```
void randomInitialTS (int index, int currentSlot, int[][] conflictcourse, int[][] timeslotrooms, int[][] suitablerooms
        int timeslot, int[][] suitableslot, int[][] afterslot, int[][] beforeslot, Random r, int low, int high){
    outerloop:
    for (int i =0; i < 100; i++){
        if (searchTS(index, (currentSlot+1), conflictcourse, suitableslot, afterslot, beforeslot)) { //kalau searchTS
            for (int k = 0; k < timeslotrooms[currentSlot].length; k++) {
                //cek contraint mengenai timeslot dan ruangan
                if (cekTimeslotRooms(index, timeslotrooms, suitablerooms, currentSlot, k)){
                    //apabila lolos cek constrain, taruh timeslot dan ruangan
                    placeTimeslotRooms(index, timeslotrooms, currentSlot, k);
                            System.out.println("Event "+ index +" ditaruh courseTimeslot " + (j+1));
                    break outerloop;
                }
            }
        }
        currentSlot= (r.nextInt(high-low) + low);
    }
}
```

Gambar 4.8 Random Initial TS

Nilai random ditetapkan dengan menetapkan slot yang paling rendah dan slot yang paling tinggi. Dari kedua nilai tersebut, bisa dipilih secara acak untuk penjadwalannya. Setelah nilai random ditetapkan, pengecekan apabila event dijadwalkan pada timeslot nilai random tersebut dilakukan. Apabila event boleh dijadwalkan pada slot tersebut dan event tidak bertabrakan dengan event yang konflik, maka event dijadwalkan pada slot nilai random tersebut.

```
//membagi range timeslot antar event untuk apabila di random
int low = k*(timeslotk / reqEvent.size());
int high = (k+1)*timeslotk / reqEvent.size();
int random = (r.nextInt(high-low) + low);
```

Gambar 4.9 Penentuan Range Random

Pengurutan event dari akhir berdasarkan slot maksimal timeslot yang ditetapkan. Berikut pseudocode urut akhir

Pseudocode 4.4 Last Sort Initial Solution

```
1: procedure Last Sort Initial Solution
2:    totaltimeslot ← 45
3:    timeslot ← matrix event and slot
4:    index ← no event
5:   indexconflict ← event conflict with index, can't be in the
     same slot
6:     suitableSlot ← matrix that contains event with
     suitableslot, event can be scheduled on timeslot if 1
7:   timeslotrooms ← matrix slot and room
8:   suitabeRoom ← matrix event and room, room is suitable for
     event if 1
9:   room ← matrix room and index
10:  repeat
11:   for i = totaltimeslot to 1 do
12:     if suitableSlot = 1 then //procedure searchTS
13:      if index timeslot!= indexconflict timeslot then
         //procedure searchTS
14:              timeslot[index] = i //procedure searchTS
15:          for j = 1 to timeslotrooms[i].length do
16:            if timeslotRooms slot i room j isEmpty then
17:              if suitableRoom event i room j isSuitable then
18:                 place room[index]=j
19:               end if
20:             else if timeslotRooms slot i room j isNotEmpty
                  then
21:               return false
22:             end if
23:         end for
24:       end if
25:     else if suitableSlot = 0 then
26:       return false
27:     end if
28:    end for
29:   until stop condition
30: end procedure
```

```
void lastSortInitialTS (int index, int currentSlot, int[][] conflictcourse, int[][] timeslotrooms, int[][] suitableroo
        int timeslot, int[][] suitableslot, int[][] afterslot, int[][] beforeslot){
    outerloop:
    for (int i = currentSlot; i > 0 ; i--){
        if (searchTS(index, (i+1), conflictcourse, suitableslot, afterslot, beforeslot)) { //kalau searchTS benar maka
            for (int k = 0; k < timeslotrooms[i].length; k++) {
                //cek contraint mengenai timeslot dan ruangan
                if (cekTimeslotRooms(index, timeslotrooms, suitablerooms, i, k)){
                    //apabila lolos cek constrain, taruh timeslot dan ruangan
                    placeTimeslotRooms(index, timeslotrooms, i, k);
                      System.out.println("Event "+ index +" ditaruh courseTimeslot " + (j+1));
                    break outerloop;
                }
            }
        }
    }
}
```

Gambar 4.10 Last Sort Initial TS

4.1.7. Pemilihan LLH dan Penjadwalan Inisiasi Awal

Event pada array awal, dipastikan bahwa event belum memiliki slot waktu sebelum melanjutkan ke proses selanjutnya. Kemudian apabila event setelahnya pada array juga belum memiliki slot waktu, maka LLH yang dipilih adalah acak. Apabila event setelahnya memiliki slot waktu, maka LLH yang dipilih adalah urut awal dengan maksimal slot waktu event setelahnya. Namun apabila setelah semua proses tersebut, event pada array awal belum mendapatkan slot waktu yang cocok maka LLH yang dipilih adalah urut awal dengan maksimal slot waktu sesuai dengan timeslot yang ditetapkan. Hal ini sesuai dengan pseudocode First Order of Array Order Slot.

Pseudocode 4.5 First Order of Array Order Slot

```
1: procedure First Order of Array Order Slot
2:    timeslot ← matrix
3:    index ← no event
4:    if timeslot[Second Order of Array Order Slot]==0 then
5:     do procedure Random Sort Initial Solution
6:    end if
7:    if timeslot[Second Order of Array Order Slot]>0 then
8:     do procedure First Sort Initial Solution
9:    end if
10:   if timeslot[index]==0 then
11:    do procedure First Sort Initial Solution
12:   end if
13: end procedure
```

```
if (courseTimeslot[arr[k]]==0){
    if (k==0){
        if (courseTimeslot[arr[k+1]]==0){
            System.out.println("event "+arr[k]);
            System.out.println("tes awal");
            randomInitialTS(arr[k], random, conflictcourse, timeslotrooms, suitablerooms,
                    timeslot, suitableslot, afterslot, beforeslot, r , low, high);
        }
        if (courseTimeslot[arr[k+1]]>0){
            int TScour = courseTimeslot[arr[k+1]];
            System.out.println("cek TS setelahnya :"+TScour);
            firstSortInitialTS(arr[k], k, conflictcourse, timeslotrooms, suitablerooms,
            TScour, suitableslot, afterslot, beforeslot);
            System.out.println("sort first");
        }
        if (courseTimeslot[arr[k]]==0){
            firstSortInitialTS(arr[k], k, conflictcourse, timeslotrooms, suitablerooms,
            timeslotk, suitableslot, afterslot, beforeslot);
            System.out.println("sort first");
        }
    }
```

Gambar 4.11 Pemilihan LLH dan Penjadwalan Inisiasi Awal

Event yang terletak pada tengah array, dicek dahulu slot waktu array event sebelumnya. Kemudian apabila nilai random lebih besar dari slot waktu array event sebelumnya dan slot waktu array event sebelumnya kurang dari maksimal timeslot yang ditetapkan, maka LLH yang dipilih adalah random. Apabila dari proses tersebut tetap belum ditemukan slot waktu yang sesuai dan slot waktu array event sebelumnya kurang dari maksimal timeslot yang ditetapkan, maka LLH yang dipilih adalah urut awal. Hal ini sesuai dengan pseudocode Middle Order of Array Order Slot.

Pseudocode 4.6 Middle Order of Array Order Slot

```
1: procedure Middle Order of Array Order Slot
2:    timeslot ← matrix
3:    index ← no event
4:    low ← event no in arrayorderSlot*(timeslot/array
arrayorderSlot size)
5:    high ← (event no in arrayorderSlot+1)*(timeslot/array
arrayorderSlot size)
6:    random ← rand(high-low) + low
7:    TScour ← timeslot[index before order]
8:    if random>TScour && TScour<45 then
9:     do procedure Random Sort Initial Solution
10:    end if
11:    if timeslot[index]==0 then
12:     do procedure First Sort Initial Solution
13:    end if
14: end procedure
```

```
if (k>0 && k<reqEvent.size()-1){
    System.out.println("tes tengah");
    int TScour = courseTimeslot[arr[k-1]];
    System.out.println("cek TS sebelumnya :"+TScour);
    if (random > TScour && TScour<timeslot) {
        randomInitialTS(arr[k], random, conflictcourse, timeslotrooms, suitablerooms,
            timeslot-TScour, suitableslot, afterslot, beforeslot, r, TScour, high);
    }
    if (courseTimeslot[arr[k]]==0 && TScour<timeslot){
        firstSortInitialTS(arr[k], TScour, conflictcourse, timeslotrooms, suitablerooms,
        timeslot-TScour, suitableslot, afterslot, beforeslot);
        System.out.println("sort first");
    }
}
```

Gambar 4.12 Initial TS Event pada Array urutan di tengah

Event yang terletak pada akhir array, dicek dahulu slot waktu array event sebelumnya. Apabila nilai random lebih besar dari slot waktu array event sebelumnya, maka LLH yang dipilih adalah acak. Apabila slot waktu event masih

kosong, dicek dahulu apakah masih ada event setelahnya. Jika tidak ada, maka LLH yang dipilih adalah urut akhir. Hal ini sesuai dengan pseudocode Last Order of Array Order Slot.

Pseudocode 4.7 Last Order of Array Order Slot

```
1: procedure Last Order of Array Order Slot
2:    timeslot ← matrix
3:    index ← no event
4:    low ← event no in arrayorderSlot*(timeslot/array
arrayorderSlot size)
5:    high ← (event no in arrayorderSlot+1)*(timeslot/array
arrayorderSlot size)
6:    random ← rand(high-low) + low
7:    TScour ← timeslot[index before order]
8:    if random>TScour && TScour<45 then
9:     do procedure Random Sort Initial Solution
10:    end if
11:    if timeslot[index]==0 then
12:     do procedure Last Sort Initial Solution
13:    end if
14: end procedure
```

```
if (k==reqEvent.size()-1){
    System.out.println("tes akhir");
    int TScour = courseTimeslot[arr[k-1]];
    System.out.println("cek TS sebelumnya :"+TScour);
    if (random > TScour) {
        randomInitialTS(arr[k], TScour, conflictcourse, timeslotrooms, suitablerooms,
        timeslotk-TScour, suitableslot, afterslot, beforeslot, r , low, high);
    }
    if (courseTimeslot[arr[k]]==0){
        if(afterslot[arr[k]].length==0){
            lastSortInitialTS(arr[k], (timeslot-2), conflictcourse, timeslotrooms, suitableroo
        timeslotk-TScour, suitableslot, afterslot, beforeslot);
            System.out.println("sort last");
        }
    }
}
```

Gambar 4.13 Initial TS Event pada Array urutan terakhir

Selanjutnya, event yang tidak memiliki urutan order dijadwalkan. LLH yang digunakan adalah urut awal.

Pseudocode 4.8 Timeslot for Event Without Order

```
1: procedure Timeslot for Event Without Order
2:    timeslot ← number of timeslot = 45
3:    index ← no event
4:     if timeslot[index]==0 then
5:       do procedure First Sort Initial Solution
6:     end if
7:   until stop condition
8: end procedure
```

```
if (courseTimeslot[index]==0){
    firstSortInitialTS(index, 0, conflictcourse, timeslotrooms, suitablerooms,
                       timeslot, suitableslot, afterslot, beforeslot);
}
```

Gambar 4.14 Initial TS Event tanpa urutan

4.1.8. Pengecekan Urutan *Event*

Pada tahap akhir, dilakukan pengecekan kembali untuk event yang memiliki urutan. tsA merupakan slot waktu event A, dan tsB merupakan slot waktu event B. Apabila nilai urutan event 1 maka seharusnya nilai tsA lebih kecil dari tsB. Jika nilai tsA lebih besar daripada nilai tsB dan nilai urutan event adalah 1, maka hal tersebut melanggar batasan urutan. Hal ini bisa diselesaikan dengan memindahkan event B ke slot waktu yang lebih besar dari tsA, atau jika tidak ada slot waktu yang sesuai maka event B tidak dijadwalkan. Saat pemindahan event B, juga harus memerhatikan urutan dari event B itu sendiri. Array ReqMoveEvent berisi urutan event B dibuat agar pemindahan slot waktu bisa dilakukan dengan benar. LLH yang dipilih adalah urut awal dengan nilai minimal slot event A. Sebelum event B dipindahkan, tentu saja slot waktu dan ruangan yang tadinya dijadwalkan untuk event B harus dikosongkan dahulu. Hal ini sesuai dengan pseudocode Check Order Event 1.

Pseudocode 4.9 Check Order Event 1

```
1: procedure Check Order Event 1
2:    tsA ← timeslot[Event A]
3:    tsB ← timeslot[Event B]
4:   beforeslot ← matrix contains event that must be scheduled
     before index
5:    afterslot ← matrix contains event that must be scheduled
      after index
6:   suitableOrder ← matrix that contains event with Order, -1
     or 1
7:    if tsA > tsB && suitableOrder[event A][event B]==1 then
8:    add Event A to Array
9:    add Event B to array, place after event A
10:     if  afterslot[Event  B].length>0  or  beforeslot[Event
        B].length>0 then
11:       repeat
12:         for k = 1 to beforeslot[Event B].length do
13:           add beforeslot[Event B][k] to Array, place before
              event A
14:         end for
15:       repeat
16:         for k = 1 to afterslot[Event B].length do
```

```
17:             add afterslot[Event B][k] to Array, place at the
                last spot Array
18:         end for
19:         tsB = 0
20:         room[Event B] = 0
21:         do procedure First Sort Initial Solution
22:           if afterslot[Event B].length>0 then
23:             add to array ReqSlotOrder
24:           end if
25:       end if
26:     end if
27: end procedure
```

```
if (tsA > tsB && suitableorder[i][j]==1) {
    System.out.println("Event "+i+" di "+tsA + ", Event "+j+" di " +tsB +",suitable order nya <0? "+ suitableorder[i][j]);
    reqMoveEvent.add(0,i);
    reqMoveEvent.add(1,j);
    if (afterslot[j].length>0 || beforeslot[j].length>0){
        for (int k = 0; k < beforeslot[j].length; k++) {
            reqMoveEvent.add(0, beforeslot[j][k]);
            }
        for (int k = 0; k < afterslot[j].length; k++) {
            reqMoveEvent.add(reqMoveEvent.size(), afterslot[j][k]);
            }
    }
    System.out.println("reqMoveEvent "+reqMoveEvent);
    moveTimeslotRooms(j, timeslotrooms, courseTimeslot[j], courseRoom[j]);
    firstSortInitialTS(j, tsA-1, conflictcourse, timeslotrooms, suitablerooms,
            timeslot-tsA, suitableslot, afterslot, beforeslot);
    System.out.println("event "+j+", pindah ke "+courseTimeslot[j]);
    if (afterslot[j].length>0){
        for (int k = 0; k < afterslot[j].length; k++) {
            reqSlotOrder.add(0, afterslot[j][k]);
            }
        System.out.println("reqSlotOrder "+reqSlotOrder);
    }
```

Gambar 4.15 Pengecekan Urutan Event (1)

Apabila event B memiliki urutan event lain setelah event B dilaksanakan, event tersebut dibuatkan array ReqSlotOrder. Apabila nilai slot waktu event B adalah 0, atau tidak dijadwalkan maka nilai urutan event lain pada ReqSlot Order juga tidak dijadwalkan. Apabila event B memiliki jadwal, maka event di ReqSlotOrder dijadwalkan menggunakan LLH urut awal dengan slot waktu minimal event B.

Aturan pemindahan yang sama juga dilakukan apabila nilai urutan adalah -1 namun nilai tsA lebih kecil daripada nilai tsB. Seharusnya jika nilai urutan event adalah -1 maka nilai tsA lebih besar dari tsB.

4.1.9. *Distance to Feasibility*

Pada dataset ITC-2007 terdapat lima *hard constraints* dimana agak mustahil jika semua harus terpenuhi dalam jangka waktu pemrosesan yang terbatas. Membiarkan sebuah event tidak dijadwalkan merupakan sebuah cara agar sebuah event tidak menyebabkan solusi tidak layak karena melanggar *hard constraints*. Distance to Feasibility adalah total siswa yang mendaftar pada event yang tidak dijadwalkan. Berikut pseudocodenya

Pseudocode 4.10 Distance to Feasibility (DTF)

```
1: procedure Distance to Feasibility (DTF)
2:    timeslot ← matrix timeslot
3:    room ← matrix room
4:    totalEvent ← total number of courses
5:    studentNumber ← number of students enrolled to the event
6:    repeat
7:    for i = 1 to totalEvent do
8:      if timeslot[i]==0 && room[i]==0 then
9:      DTF = DTF + studentNumber[i]
10:    end for
11: end procedure
```

```
//Distance to feasibility. cek event yang tidak ditempatkan dan berapa siswanya
int distFeasibility(int[] sizeStuEv){
    distFeasibility = 0;
    noTimeslot = new ArrayList<>();
    for (int i = 0; i < courseTimeslot.length; i++) {
        if (courseTimeslot[i] < 1 || courseRoom[i] < 1) {
            noTimeslot.add(i);
            distFeasibility= distFeasibility + sizeStuEv[i];
        }
    }
    double prosentaseTotCourse = (noTimeslot.size()*100)/courseTimeslot.length;
    System.out.println("distFeasibility"+distFeasibility);
    System.out.println("prosentase Feasibility"+(100-prosentaseTotCourse));
    return distFeasibility;
}
```

Gambar 4.16 Distance to Feasibility

4.1.10 *Precedence Violations* / Pelanggaran urutan

Pada event yang memiliki urutan, event yang boleh tidak ditempatkan tentu saja event yang urutannya di akhir. Dengan kata lain, event yang harus ditempatkan sebelum urutan event lain harus dijadwalkan terlebih dahulu. Hal ini sudah diantisipasi pada pengecekan urutan event yang dibahas pada sub-bab 5.1.8. Penghitungan pelanggaran urutan diperlukan untuk menentukan apakah proses

pembentukan solusi awal perlu diulang dan berapa kali diulang agar solusi awal bisa memenuhi *hard constraints* yang ada.

Pseudocode 4.11 Precedence Violations

```
1: procedure Precedence Violations
2:    tsA ← timeslot[Event A]
3:    tsB ← timeslot[Event B]
4:    totalEvent ← total number of courses
5:    suitableOrder ← matrix that contains event with Order, -1
      or 1
6:    repeat
7:      for i = 0 to totalEvent do
8:       if tsA > 0 then
9:        repeat
10:          for j = 1 to totalEvent do
11:            if tsB > 0 then
12:              if tsA > tsB && suitableOrder[event A][event
                 B]==1 then
13:                 PrecedenceViolations++
14:              end if
15:              if tsA < tsB && suitableOrder[event A][event
                 B]==-1 then
16:                 PrecedenceViolations++
17:              end if
18:            end if
19:          end for
20:       end if
21:      end for
22:  end procedure
```

```
int precedenceViolation(int[][] suitableorder){
    precedenceViolation=0;
    for (int i = 0; i < courseTimeslot.length; i++) {
        int tsA=courseTimeslot[i];
        if (tsA>0){
            for (int j = i+1; j < courseTimeslot.length; j++) {
                int tsB = courseTimeslot[j];
                if (tsB>0){
                    if (tsA > tsB && suitableorder[i][j]==1) {
                          System.out.println("Event "+i+" di "+tsA + ", Event "+j+" di " +tsB +",suitable order nya <0
                        precedenceViolation++;
                    }
                    if (tsA < tsB && suitableorder[i][j]==-1) {
                          System.out.println("Event "+i+" di "+tsA + ", Event "+j+" di " +tsB +",suitable order nya >0
                        precedenceViolation++;
                    }
                }
            }
        }
    }
      System.out.println("precedenceViolation "+precedenceViolation);
    return precedenceViolation;
}
```

Gambar 4.17 Pelanggaran urutan

4.2. Proses Optimasi

Pada proses optimasi, algoritma *Self-Adaptive* dengan *Tabu Search*, dan *Simulated Annealing* digunakan. Algoritma *Self-Adaptive* digunakan sebagai algoritma seleksi pemilihan *Low Level Heuristic* (LLH) yang digunakan. Algoritma *Tabu Search* dan *Simulated Annealing* digunakan sebagai *Move Acceptance* pada penelitian ini. Implementasi dari ketiga algoritma tersebut dibahas pada subbab berikut

4.2.1. Seleksi Pemilihan LLH dengan Self-Adaptive

```
//index NL Self-Adaptive [BARU]
NL = new int[totalNL];
// generate random numbers within 1 to 10
for (int k=0; k< totalNL; k++) {
    int rand = (int)(Math.random() * range) + minrand;
    // Output is different everytime this code is executed
    NL[k]=rand;
      System.out.println("isi array " + rand);
}

//WNL Self-Adaptive [BARU]
LinkedList WNL = new LinkedList();

readInitialSol(mStudentEvent, suitableRoom,
        suitableSlot, suitableOrder,
        timeslot, initialTS, initialRoom);
high = currentTimeslot.length;
```

Gambar 4.18 Pembentukan NL, WNL, dan Pembacaan Initial Sol

Pada strategi Self Adaptive, *Neighboring List* (NL) sejumlah *k* yang telah ditetapkan dibentuk terlebih dahulu. NL berisi random nilai LLH yang telah ditetapkan. *Winning Neighboring List* (WNL) dibentuk untuk menyimpan LLH yang dapat menghasilkan solusi yang lebih baik.

```
do {
    //menghilangkan nilai WNL
    WNL.clear();
    newRoom = currentRoom.clone();
    newTimeslotRooms = currentTimeslotRooms.clone();
```

Gambar 4.19 WNL, NewRoom, dan NewTimeslotRoom

Pada setiap iterasi NL selesai, WNL akan dipindah ke NL sehingga WNL kosong. Sehingga di awal dipastikan bahwa WNL kosong terlebih dahulu. *newRoom* dan *newTimeslotRooms* berfungsi untuk menyimpan ruangan dan penanda slot-ruangan yang digunakan.

Setiap iterasi NL, memiliki 4 LLH yang digunakan yaitu Move 1, Move 2, Swap 2, dan Swap 3. Pada LLH Move 1, LLH dapat mengganti ruangan event tersebut sehingga ketika dipindah ruangan bisa menyesuaikan dengan ruangan yang tersedia.

Pseudocode 4.12 SelfAdaptive NL

```
1: procedure SelfAdaptive NL
2:   newTimeslot ← matrix timeslot
3:   newRoom ← matrix room
3:   repeat
4:   for k = 0 to totalNL do
5:      newRoom ← currentRoom.clone
6:      newTimeslotRooms ← currentTimeslotRooms.clone
7:      switch
8:         case 1 do
9:           newTimeslot = move1TS //LLH move1 timeslot
10:        break
11:        case 2 do
12:          newTimeslot = swap2TS //LLH swap2 timeslot
13:        break
14:        case 3 do
15:          newTimeslot = move2TS //LLH move2 timeslot
16:        break
17:        case 4 do
18:          newTimeslot = swap3TS //LLH swap3 timeslot
19:        break
20:     end switch
21:   end for
22: end procedure
```

```
for (int k=0; k< totalNL; k++) {
    newRoom = currentRoom.clone();
    newTimeslotRooms = currentTimeslotRooms.clone();
    switch(NL[k]) {
      case 1: newTimeslot = move1Ts(currentTimeslot, timeslot);
        break;
      case 2: newTimeslot = swap2Ts(currentTimeslot);
        break;
      case 3: newTimeslot = move2Ts(currentTimeslot, timeslot);
        break;
      case 4: newTimeslot = swap3Ts(currentTimeslot);
        break;
      case 5: newTimeslot = move3Ts(currentTimeslot, timeslot); break;
      case 6: newTimeslot = swap4Ts(currentTimeslot); break;
      default: newTimeslot = currentTimeslot.clone(); break;
    }
```

Gambar 4.20 Iterasi NL dan LLH

4.2.2. Penjadwalan Event yang Belum Memiliki Jadwal

Seperti yang dibahas sebelumnya, pada *initial solution* ada kemungkinan mata kuliah yang tidak memiliki timeslot atau belum ditempatkan pada

timetabling. Sehingga apabila sedang proses optimasi, mata kuliah tersebut bisa ditempatkan dengan memperhatikan *Hard Constraints* yang ada. Hal ini dijelaskan pada pseudocode *Input Event Unscheduled While Optimize*.

Pseudocode 4.13 Input Event Unscheduled While Optimize

```
1: procedure Input Event Unscheduled While Optimize
2:   newTimeslot ← matrix timeslot
3:   newRoom ← matrix room
4:   newTimeslotrooms ← matrix timeslot and room
3:   repeat
4:   for j = 0 to totalEvent do
5:      if (timeslot[j]==0 then
6:        newTimeslot = procedure First Sort Optimization
7:        if newTimeslot[j]>0
8:          newRoom = procedure SearchRoom
8:         case 1 do
9:           newTimeslot = move1TS //LLH move1 timeslot
10:         break
11:         case 2 do
12:           newTimeslot = swap2TS //LLH swap2 timeslot
13:         break
14:        case 3 do
15:           newTimeslot = move2TS //LLH move2 timeslot
16:         break
17:         case 4 do
18:           newTimeslot = swap3TS //LLH swap3 timeslot
19:         break
20:      end switch
21:   end for
22: end procedure
```

```
// input course yang tidak memiliki timeslot sebelumnya
for (int j=0; j<conflictCourse.length; j++){
    if (currentTimeslot[j]==0){
        newTimeslot = firstSortOptimizationTS(j, 0, conflictCourse, timeslotrooms, suitableRoom,
                      timeslot, suitableSlot, afterSlot, beforeSlot);
        if (newTimeslot[j]>0)
        newRoom = searchRoom(j, newTimeslot[j]-1, suitableRoom);
        if (newTimeslot[j]>0 && newRoom[j]>0)
        newTimeslotRooms = placeTimeslotRooms(j, newTimeslotRooms, newTimeslot[j]-1, newRoom[j]-1);
    }
}
```

Gambar 4.21 Input Event yang belum memiliki timeslot

4.2.3. Perhitungan nilai penalty dan nilai DTF baru

Pada gambar 4.2.2 schStudent merupakan perhitungan siswa menghadiri event apa saja. Hal ini berkaitan dengan perhitungan pelanggaran *soft constraints.* Kemudian menggunakan timeslot dan ruangan yang telah dioptimasi, hard

constraintsnya dicek dahulu apakah telah memenuhi. Apabila solusi optimasi memenuhi *hard constraints*, maka penalti dan *distance to feasibility* dihitung. Apabila tidak memenuhi, maka penalty berjumlah M yaitu 1000000.

```
int[][] schStudent = readSol.studentAvail(mStudentEvent, timeslot, newTimeslot);
hardConstraint = checkHC.hardConstraint(schStudent, suitableRoom,
        suitableSlot, suitableOrder,
        newRoom, newTimeslot);
if (hardConstraint) {
    newPenalty = checkPenalti.totalPenalti(schStudent);
    newDistFeasibility = checkPenalti.distFeasibility(newTimeslot, newRoom, sizeeventstudent,
    System.out.println("newDistFeasibility " +newDistFeasibility);
} else {
    newPenalty = M;
}
```

Gambar 4.22 Penghitungan nilai penalti dan nilai distance to feasibility baru

4.2.4. Move Acceptance dengan Simulated Annealing dan Tabu Search

Perhitungan penalti iterasi sebelumnya dengan perhitungan penalti iterasi baru dihitung pada gambar 4.23. Begitu juga dengan *distance to feasibility*. Perhitungan *distance to feasibility* mencegah semakin bertambah. Apabila iterasi baru memiliki nilai yang lebih baik pada nilai penalti maupun distance to feasibility maka solusi tersebut diterima dan LLH nya disimpan pada WNL. Pseudocode mengenai Move Acceptance dapat dilihat sebagai berikut

Pseudocode 4.14 Move Acceptance

```
1: procedure Move Acceptance
2:   delta ← current penalty – new penalty
3:   deltaDTF ← current DTF – new DTF
4:   WNL ← array winning neighboring list 2:
5:   currrentTimeslot, newTimeslot ← matrix timeslot
6:   currentRoom, newRoom ← matrix room
7:   currentTimeslotrooms, newTimeslotrooms ← matrix timeslot
    and room
8:   if delta>0 && deltaDTF >= 0 && newpenalty!=1000000 then
9:      accept solution
10:     update currentTimeslot, currentRoom, currentPenalty,
      currentTimeslotRooms, DTF
11:     add LLH to WNL
12:   else if deltaDTF>=0 then
13:    count BoltzmanEquation
14:    if BoltzmanEquation>random then
15:      if TabuList contains newTimeslot then
16:        if tabusize >= tabulistlength then
17:          remove one list from tabu list
18:          add newTimeslot on the tabu list
19:        else
20:          add newTimeslot on the tabu list
```

```
21:          end if
22:       else
23:          accept solution
24:          update currentTimeslot, currentRoom, currentPenalty,
             currentTimeslotRooms, DTF
25:          if TabuList contains newTimeslot then
26:             if tabusize >= tabulistlength then
27:               remove one list from tabu list
28:               add newTimeslot on the tabu list
29:             else
30:               add newTimeslot on the tabu list
31:             end if
32:          end if
33:       end if
34:     end if
35:  end if
36: end procedure
```

```
delta = currentPenalty - newPenalty;
deltaDistFeasibility = distFeasibility - newDistFeasibility;

  System.out.println("Current Penalty Score: " + currentPenalty);
  System.out.println("New Penalty Score: " + newPenalty);
if (delta > 0 && deltaDistFeasibility >= 0 && newPenalty != M) {
    currentTimeslot = newTimeslot.clone();
    currentRoom = newRoom.clone();
    currentPenalty = newPenalty;
    currentTimeslotRooms=newTimeslotRooms.clone();
    distFeasibility = newDistFeasibility;
    System.out.println("Accept Solution");
    //add to WNL
    WNL.add(NL[k]);
```

Gambar 4.23 Move Acceptance

Apabila nilai distance to feasibility tidak memburuk, maka perhitungan Simulated Annealing digunakan seperti pada gambar 4.24. Apabila perhitungan boltzman lebih besar dari random nilai yang dihasilkan dan solusi tidak terdapat pada *tabu list*, maka solusi iterasi diterima.

```
} else if (deltaDistFeasibility >= 0){
    p = Math.pow(Math.E, (-(Math.abs(delta)/T)));
    if (p > r.nextDouble()) {
        if (tabu.contains(newTimeslot)) {
            if (tabu.size() >= tabulistLength) {
                tabu.pop();
                tabu.push(newTimeslot);
            } else {
                tabu.push(newTimeslot);
            }
        }
        else {
        currentTimeslot = newTimeslot.clone();
        currentRoom = newRoom.clone();
        currentPenalty = newPenalty;
        currentTimeslotRooms=newTimeslotRooms.clone();
        distFeasibility = newDistFeasibility;
        System.out.println("Accept Solution");
        if (tabu.size() >= tabulistLength) {
                tabu.pop();
                tabu.push(currentTimeslot);
            } else {
                tabu.push(currentTimeslot);
            }
        WNL.add(NL[k]);
```

Gambar 4.24 Simulated Annealing and Tabu List

```
if (n%Tchange == 0) {
    T = T * alpa;
      System.out.println("suhu alpa "+T);
}
if (n%Nreheating == 0) {
    T = T + (T*beta);
      System.out.println("suhu beta"+T);
}
n++;

}
```

Gambar 4.25 Nilai suhu simulated annealing

Nilai suhu alpa dan beta yang digunakan dalam perhitungan Boltzman Simulated Annealing dapat dilihat pada gambar 4.25.

4.2.5. Memindahkan WNL ke NL Self-Adaptive

Langkah terakhir dalam optimasi, yaitu mengosongkan NL kemudian memindahkan value dari WNL ke NL kembali. NL diisi dengan 75% dari WNL, sisanya 25% diisi dengan random nilai LLH. Apabila WNL tidak terisi setelah iterasi sebelumnya, maka keseluruhan NL diisi dengan nilai random.

Pseudocode 4.15 Move WNL to NL

```
1: procedure Move WNL to NL
2:    WNLsize ← total count WNL
3:    NL ← array for storing choosen LLH
4:    totalNL ← total NL
5:   WNL ← array for storing LLH that successfully made better
      solution
```

```
6:   totalLLH ← total LLH
7:   ulangWNL ← adding WNL to NL randomly
8:    NLbaru ← 75% NL
9:   repeat
10:   for i = 1 to totalNL do
11:     if WNL isEmpty then
12:      NL = random*totalLLH
13:     else if NLbaru > i //i is 75% NL
14:      if WNLsize > i
15:        NL[i] = WNL[i] //add WNL to NL
16:      else
17:        ulangWNL = random*WNLsize
18:       NL[i] = WNL[ulangWNL] //add random WNL to NL
19:      end if
20:     else // i is 25% NL
21:      NL = random*totalLLH
22:     end if
23:   end for
24: end procedure
```

```
//Empty NL
Arrays.fill(NL, 0);
  System.out.println("List WNL " + WNL);
int WNLsize= WNL.size();
maxrand = WNLsize;
  System.out.println("size WNL "+WNL.size());
//75% element NL
int NLbaru = Math.abs((3*totalNL)/4);
  System.out.println("NLbaru " + NLbaru);
for (int j=0; j<= totalNL-1; j++) {
   //apabila WNL kosong, NL dirandom lagi
   if (WNLsize == 0){
      int rand = (int)(Math.random() * range) + minrand;
      NL[j]=rand;
   }
```

```
   //75% dari NL diisi dengan WNL
   else if(NLbaru > j){
      if (WNLsize > j) {
         NL[j]=(int)WNL.get(j);
           System.out.println("lolos " + WNL.get(j));
      }
      else{
         int ulangWNL = (int)Math.random()*WNLsize;
         NL[j]=(int)WNL.get(ulangWNL);
         System.out.println("cek "+  NL[j]);
      }
   }
   //25% dari NL diisi random
   else{
      int rand = (int)(Math.random() * range) + minrand;
      NL[j]=rand;
      }
```

Gambar 4.26 Pemindahan WNL ke NL

Iterasi diatas dilakukan kembali sampai penalti bisa mencapai angka 0 atau waktu yang ditetapkan sudah habis. Apabila stopping condition tercapai, maka solusi yang ada disimpan dan nilai akhir ditampilkan.

BAB 5
UJI COBA DAN ANALISIS HASIL

5.1. Data Uji Coba

Data uji coba yang digunakan pada penelitian ini adalah dataset ITC-2007 dan dataset Socha. Dataset Socha memiliki 11 instances. Sedangkan dataset ITC-2007 memiliki 24 instances.

5.2. Lingkungan Uji Coba

Spesifikasi perangkat keras yang digunakan yaitu

Tabel 5.1 Spesifikasi Perangkat Keras

Perangkat Keras	Spesifikasi
Jenis	Laptop Dell Inspiron 14 7000 Series
Processor	Intel(R) Core i7-4720 HQ 2.6 GHz
RAM	8 GB
Hard Disk Drive	1000 GB

Spesifikasi perangkat lunaknya yaitu

Tabel 5.2 Spesifikasi Perangkat Lunak

Perangkat Lunak	Fungsi
Windows 10 64 bit	Sistem Operasi
Netbeans 8.2	Implementasi Algoritma
Notepad++	Pengolahan Data dan Hasil
Microsoft Excel	Pengolahan Hasil Uji Coba
Microsoft Word	Penulisan Laporan

5.3. Algoritma *Tabu-Simulated Annealing*

Sebelum dilakukan uji coba, algoritma *Tabu-Simulated Annealing* telah dilakukan sebagai acuan dalam pengembangan aplikasi. Algoritma *Tabu-Simulated Annealing* diuji cobakan pada kedua dataset yaitu dataset socha dan dataset ITC2007. Algoritma yang digunakan yaitu algoritma *Tabu-Simulated Annealing* yang telah dikembangkan oleh Ahsanul.

5.3.1. Skenario Algoritma *Tabu-Simulated Annealing*

Parameter yang digunakan pada skenario Algoritma *Tabu-Simulated Annealing* dapat dilihat pada tabel 5.3. Pada tabel 5.4. dapat dilihat daftar skenario yang diuji coba pada Algoritma *Tabu-Simulated Annealing*.

Tabel 5.3 Keterangan Parameter Skenario Algoritma *Tabu-Simulated Annealing*

Parameter	Keterangan	Algoritma
LLH	Jumlah *low level heuristic* yang digunakan	
T_0	Suhu awal algoritma Simulated Annealing	*Simulated Annealing*
T_1	Suhu akhir algoritma Simulated Annealing	*Simulated Annealing*
α	Penurunan suhu	*Simulated Annealing*
Nα	Jumlah iterasi setiap penurunan suhu	*Simulated Annealing*
β	Kenaikan suhu pada proses reheating	*Simulated Annealing*
Nβ	Jumlah iterasi setiap kenaikan suhu (*reheating*)	*Simulated Annealing*
TL	Panjang *tabu list* solusi algoritma Tabu Search	*Tabu Search*
TLLH	Panjang *tabu list - low level heuristics*	*Tabu Search*
RW	Metode *roulette Wheel* yang digunakan	

Skenario yang dilakukan yaitu

Tabel 5.4 Skenario Algoritma *Tabu-Simulated Annealing*

Skenario	Iterasi (juta)	Time Limit	LLH	T_0	T_1	α	Nα	β	Nβ	TL	TLLH	RW
K	3	-	2	95	0	0,999	50	0,5	25000	3	-	-
N	3-6-20	-	2	95	0	0,999	50	0,5	25000	3	-	Probability

5.3.2. Hasil Eksperimen Dataset Socha

Hasil eksperimen dataset Socha akan dijelaskan dalam tabel yang berisi nama *instance*, rata-rata solusi awal (Avg Initial Sol), nilai terbaik (Best), nilai terburuk (*Worst*), rata-rata nilai (*Avg*), dan standar deviasi (St.dev). Kompleksitas masing-masing instance dataset telah dijelaskan pada sub bab 2.2.3. PE-CTT Pada Dataset Socha. Hasil eksperimen dijelaskan pada masing-masing skenario. Pada tabel 5.5. terdapat Keterangan Tabel Hasil Skenario Algoritma *Tabu-Simulated Annealing*.

Tabel 5.5 Keterangan Tabel Hasil Skenario Algoritma *Tabu-Simulated Annealing*

Kode	Keterangan
Instance	Nama Instance yang diuji coba dalam Algoritma Tabu-Simulated Annealing
Avg Initial Sol	Rata-rata solusi awal. Solusi awal didapat ketika awal proses optimasi untuk pembentukan nilai solusi penjadwalan mata kuliah
Best	Nilai penalti / solusi akhir yang paling baik yang didapatkan
Worst	Nilai penalti / solusi akhir yang paling buruk yang didapatkan
Avg	Nilai rata-rata penalti / solusi akhir yang didapatkan
St. dev	Standar deviasi dari nilai penalti / solusi akhir yang didapatkan

5.3.2.1. Hasil Skenario *Timelimit* K

Pada Skenario *timelimit* K, parameter yang digunakan sesuai dengan Tabel 5.4. Hasil eksperimen ditunjukkan pada Tabel 5.6. Pada tabel 5.6 menunjukkan bahwa semua *instance small* telah mendapatkan nilai penalti terbaik 0, sementara pada *instance medium* nilai penalti terbaik berkisar pada angka 120-246, dan *instance large* mendapatkan nilai penalti terbaik sebesar 1015. Keterangan mengenai tabel hasil skenario dapat dilihat pada tabel 5.5.

Tabel 5.6 Hasil Skenario Timelimit K (Ahsanul, 2018)

Instance	Avg Initial Sol	Best	Worst	Avg	St.dev
small1	311,7	0	2	0,6	0,67
small2	323,9	0	1	0,5	0,52
small3	291,6	0	3	1,1	0,94

Instance	Avg Initial Sol	Best	Worst	Avg	St.dev
small4	168,8	0	4	1,8	1,33
small5	454,2	0	2	0,5	0,82
medium1	1040,7	216	267	244	14,88
medium2	1039,5	210	262	246,5	14,42
medium3	1063,1	246	300	275	17,33
medium4	1079	193	245	221	16,43
medium5	1158	120	174	150	19,87
large	1971	1015	1171	1089	48,56

5.3.2.2. Hasil skenario Timelimit N

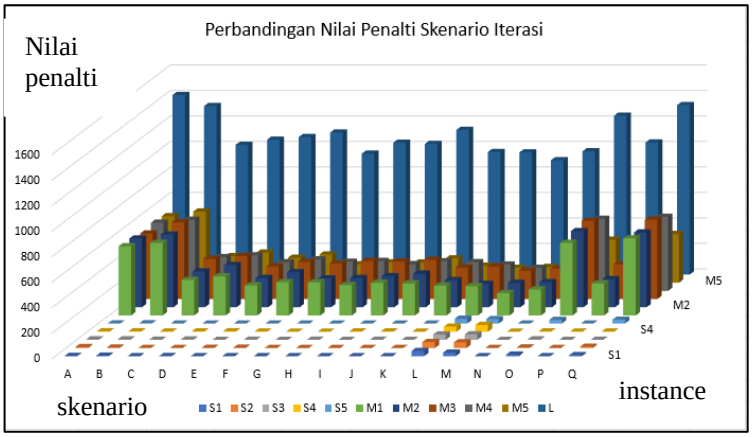
Pada Skenario *timelimit* N, parameter yang digunakan sesuai dengan Tabel 5.4. Hasil eksperimen ditunjukkan pada Tabel 5.7. Pada tabel 5.7 menunjukkan bahwa semua *instance small* telah mendapatkan nilai penalti terbaik 0, sementara pada *instance medium* nilai penalti terbaik berkisar pada angka 116-208, dan *instance large* mendapatkan nilai penalti terbaik sebesar 936. Keterangan mengenai tabel hasil skenario dapat dilihat pada tabel 5.5.

Tabel 5.7 Hasil Skenario *Timelimit* N (Ahsanul, 2018)

Instance	Avg Initial Sol	Best	Worst	Avg	St.dev
small1	315,3	0	2	0,9	0,83
small2	327,7	0	3	1,5	1,21
small3	297,6	0	7	1,8	1,89
small4	164,5	0	8	2,2	2,18
small5	454,6	0	1	0,1	0,30
medium1	1028,6	198	256	230,9	20,52
medium2	1045,8	195	268	235,5	20,53
medium3	1071,2	208	299	271,3	25,85
medium4	1069,6	181	242	219,7	21,55
medium5	1169,5	116	209	151,2	25,15
large	1960	936	1169	1048	74,55

5.3.2.3. Rekapitulasi Hasil Skenario

Setelah dilakukan uji coba skenario A hingga skenario Q, didapatkan hasil solusi masing-masing skenario. Dari keseluruhan hasil solusi skenario, dapat dibandingkan untuk pemilihan nilai solusi yang paling optimum. Perbandingan nilai penalti menggunakan nilai terbaik dari setiap instance pada setiap skenario eksperimen. Perbandingan nilai penalti eksperimen ditunjukkan pada Gambar 5.1.



Gambar 5.1 Perbandingan Nilai Penalti (Ahsanul, 2018)

Pada perbandingan tersebut, Skenario K dan Skenario N merupakan eksperimen dengan solusi yang lebih optimum daripada eksperimen lain. Sehingga kedua ekperimen ini akan diuji coba menggunakan skenario dengan timelimit.

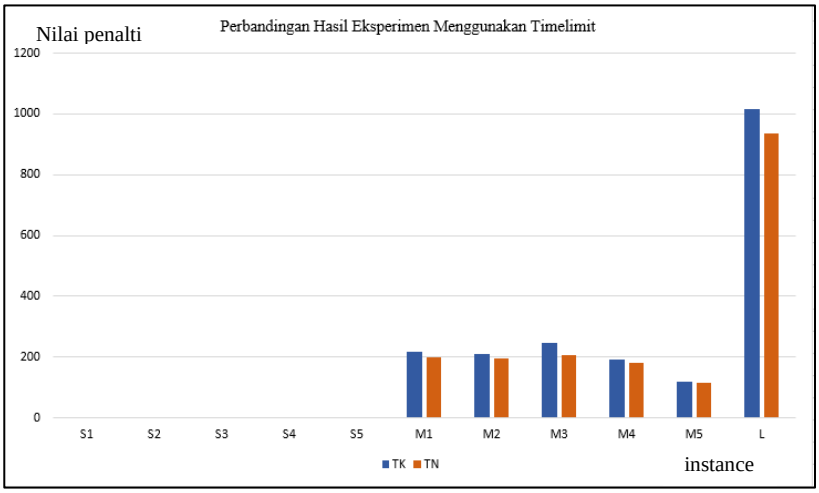
5.3.3. Perbandingan Hasil Eksperimen Dataset Socha

Perbandingan hasil eksperimen penelitian dilakukan terhadap hasil yang telah didapatkan. Perbandingan dilakukan pada hasil skenario yang paling optimum yaitu skenario K dan skenario N. Tabel 5.8 dan gambar 5.2 menunjukkan perbandingan angka terbaik dan angka rata-rata dari kedua skenario tersebut. Kolom Instance merupakan nama instance yang diuji cobakan. Kolom Best merupakan nilai penalti / solusi akhir yang paling baik yang didapatkan pada masing-masing skenario. Kolom Average merupakan nilai rata-rata penalti / solusi akhir yang paling baik yang didapatkan pada masing-masing scenario.

Tabel 5.8 Perbandingan Skenario Timelimit K dan N (Ahsanul, 2018)

Instance	Skenario Timelimit K		Skenario Timelimit N	
	Best	Average	Best	Average
small1	0	**0,6**	**0**	0,9
small2	0	**0,5**	**0**	1,5
small3	0	**1,1**	**0**	1,8
small4	0	**1,8**	**0**	2,2
small5	0	0,5	**0**	**0,1**
medium1	216	244	**198**	**230,9**
medium2	210	246,5	**195**	**235,5**

Instance	Skenario Timelimit K		Skenario Timelimit N	
	Best	Average	Best	Average
medium3	246	275	**208**	**271,3**
medium4	193	221	**181**	**219,7**
medium5	120	**150**	**116**	151,2
large	1015	1089	**936**	**1048**



Gambar 5.2 Perbandingan Hasil Eksperimen Skenario Timelimit K dan N (Ahsanul, 2018)

Skenario N dengan batasan waktu menghasilkan nilai penalti yang lebih baik daripada skenario K ditunjukkan pada tabel dan gambar. Pada skenario N, nilai penalti terbaik pada semua *instance* lebih rendah atau sama dengan skenario K. Sedangkan nilai rata-rata penalti skenario K hanya lebih unggul pada *small1*, *small2*, *small3*, *small4*, dan *small5*. Nilai rerata penalti skenario N lebih optimum pada *small5*, *medium1*, *medium2*, *medium3*, *medium4*, dan *large*.

5.3.4. Hasil Eksperimen Dataset ITC2007

Pada subbab ini membahas mengenai hasil eksperimen dataset ITC2007. Eksperimen dengan dataset ITC2007 sebelumnya tidak pernah dilakukan menggunakan aplikasi yang telah dikembangkan di Algoritma *Tabu-Simulated Annealing*. Eksperimen yang akan dilakukan menggunakan parameter yang menghasilkan solusi terbaik pada eksperimen dataset socha yaitu eksperimen N dengan batas waktu yang telah ditentukan yaitu 500 detik atau sekitar 8 menit 20 detik sesuai dengan aturan ITC2007. Dengan adanya eksperimen dataset ITC2007

diharapkan dapat membuat aplikasi mencapai hasil yang lebih optimum dan aplikasi menjadi lebih *general* dalam *Post Enrolment Course Timetabling*. Kompleksitas mengenai dataset ITC2007 dapat dilihat pada subbab 2.2.2. PE-CTT Pada Dataset ITC2007. Tabel 5.9 menunjukkan hasil eksperimen dataset ITC dengan aplikasi riset yang sebelumnya dikembangkan. Keterangan mengenai tabel hasil skenario dapat dilihat pada tabel 5.5.

Tabel 5.9 Hasil Skenario Uji Coba Dataset ITC-2007

Instance	Avg Initial Sol	Best	Worst	Avg	St.dev
early1	2917,2	2448	2692	2591,2	72,38
early2	*initial solution* lebih dari 500s, 120m				
early3	*initial solution* lebih dari 500s, 120m				
early4	*initial solution* lebih dari 500s, 120m				
early5	1886,2	1192	1460	1286,8	68,65
early6	1799,5	1140	1377	1303,4	68,08
early7	1665,9	954	1039	1003	29,82
early8	1758,8	418	660	535,9	78,74
late9	*initial solution* lebih dari 500s, 120m				
late10	*initial solution* lebih dari 500s, 120m				
late11	*initial solution* lebih dari 500s, 120m				
late12	*initial solution* lebih dari 500s, 120m				
late13	1777,1	1326	1486	1395,8	58,35
late14	*initial solution* lebih dari 500s, 120m				
late15	1603,2	420	613	488,6	52
late16	1767,6	150	258	227	48,22
late17	4336,9	0	40	20,1	12,79
late18	3162,3	1738	2140	1965,2	126,58
hidden19	*initial solution* lebih dari 500s, 120m				
hidden20	3269,5	1791	2286	2021,4	127,13
hidden21	1980,4	1095	1296	1237,9	59,83
hidden22	*initial solution* lebih dari 500s, 120m				
hidden23	*initial solution* lebih dari 500s, 120m				
hidden24	3360,2	1658	2076	1863,8	123,89

5.3.5. Perbandingan Hasil Eksperimen Dataset ITC-2007

Perbandingan hasil eksperimen penelitian dataset ITC-2007 dilakukan terhadap hasil yang telah didapatkan. Dapat dilihat pada tabel 5.10, terdapat 13 instances dari 24 instances yang solusinya berhasil tampil, yaitu *early1, early5-early8, late13, late15-late18, hidden20, hidden21,* dan *hidden24.* Sisanya yaitu 11

instances bahkan membutuhkan waktu yang sangat lama untuk memberikan solusi awal. Enam dari 11 instances yang memerlukan waktu sangat lama untuk memberikan solusi awal berisi 1000 mahasiswa, sementara empat *instances* berisi 500 mahasiswa dan satu *instance* berisi 300 mahasiswa. *Instance* dengan *events* terbanyak berjumlah 600 juga memerlukan waktu sangat lama. Namun *instance* dengan 500 *events* dapat menampilkan solusi akhir. Sehingga tidak dapat disimpulkan bahwa banyaknya jumlah *events*, *rooms*, *features*, dan *students* mempengaruhi waktu pembentukan solusi awal, namun komplikasi dari semua data tersebut bisa mempengaruhi pembentukan solusi awal. Pada kolom output dengan isian **"V"** menandakan bahwa solusi akhir dapat tampil. Sedangkan kolom output dengan isian **"X"** menandakan bahwa solusi akhir tidak dapat tampil / membutuhkan waktu yang lebih lama dari 10 menit.

Tabel 5.10 Perbandingan Hasil Eksperimen Dataset ITC-2007

Problem Instances	Time Slots	Events	Rooms	Features	Students	Output
early1	45	400	10	10	500	V
early2	45	400	10	10	500	**X**
early3	45	200	20	10	1000	**X**
early4	45	200	20	10	1000	**X**
early5	45	400	20	20	300	V
early6	45	400	20	20	300	V
early7	45	200	20	20	500	V
early8	45	200	20	20	500	V
late9	45	400	10	20	500	**X**
late10	45	400	10	20	500	**X**
late11	45	200	10	10	1000	**X**
late12	45	200	10	10	1000	**X**
late13	45	400	20	10	300	V
late14	45	400	20	10	300	**X**
late15	45	200	10	20	500	V
late16	45	200	10	20	500	V
late17	45	100	10	10	500	V
late18	45	200	10	10	500	V
hidden19	45	300	10	10	1000	**X**
hidden20	45	400	10	10	1000	V
hidden21	45	500	20	20	300	V
hidden22	45	600	20	20	500	**X**
hidden23	45	400	20	30	1000	**X**
hidden24	45	400	20	30	1000	V

5.3.6. Kesimpulan

Kesimpulan yang didapatkan dari uji coba dataset Socha dan dataset ITC 2007 ini ialah bahwa

1. Algoritma yang digunakan pada aplikasi yang telah dikembangkan yaitu *Tabu Search – Simulated Annealing* berbasis *Hyper-Heuristics* mampu menyelesaikan permasalahan penjadwalan pada *domain Post-Enrolment Course Timetabling* pada dataset Socha dan dataset ITC-2007. Meskipun begitu, aplikasi memerlukan beberapa penyesuaian ke depannya agar dataset ITC-2007 dapat dijalankan semuanya.
2. Pengubahan parameter pada skenario uji coba juga dapat mempengaruhi hasil solusi yang lebih optimum. Parameter yang dapat diubah dalam penelitian *Tabu Search – Simulated Annealing* yaitu koefisien suhu, *reheating*, *roulette wheels*, dan Tabu LLH.
3. Pada Dataset Socha, algoritma *Tabu-Simulated Annealing Hyper-heuristics* dengan pengembangan mampu mereduksi nilai penalti sebesar 40-80% dari solusi awal. Diharapkan algoritma *Tabu-Simulated Annealing-Self Adaptive Hyper-heuristics* juga mampu mereduksi nilai penalti.
4. Pada dataset ITC-2007 hasil pembentukan solusi awal membutuhkan waktu lebih lama dari batasan waktu terjadi pada *instances early2, early3, early4, late9, late10, late11, late12, late14, hidden19, hidden22, dan hidden23.* Total *instances* yang mengalami pembentukan solusi awal terlalu lama adalah 11 *instances*, sedangkan terdapat 13 *instances* berhasil membentuk solusi awal dan keluaran.

5.3.7. Saran

Saran terhadap Algoritma *Tabu-Simulated Annealing* yang telah dilakukan untuk dikembangkan ke depannya yaitu :

1. Menyesuaikan masukan format dataset ITC-2007 pada aplikasi dikarenakan terdapat perbedaan masukan format dataset Socha dan dataset ITC-2007.

2. Penambahan *hard constraints* pada dataset ITC-2007.
3. Perbaikan pembentukan solusi awal untuk dataset ITC-2007 agar dapat berhasil membentuk solusi awal dan keluaran.

Dapat dikatakan bahwa permasalahan yang dihadapi saat melakukan eksperimen dengan dataset ITC2007 yaitu pembentukan solusi awal yang melebihi batas waktu dan keluaran menghasilkan pesan *error*. Oleh karena itu, pengembangan penelitian yang akan dilakukan yaitu mengenai solusi awal yang harus diperbaiki agar dapat berjalan dengan baik dan proses optimasi yang akan menghasilkan keluaran yang layak. Selain kedua hal tersebut, perlu diingat bahwa terdapat *hard constraint* yang berbeda pada dataset ITC-2007.

5.4. Skenario Uji Coba

Parameter dan algoritma yang digunakan pada skenario uji coba eksperimen dapat dilihat pada tabel 5.11. Pada tabel 5.12. dapat dilihat daftar scenario yang diuji coba pada Algoritma Self Adaptive-Tabu-Simulated Annealing.

Tabel 5.11 Keterangan Parameter Skenario Uji Coba

Parameter	Keterangan	Algoritma
NL	Jumlah *Neighboring List* yang digunakan	*Self Adaptive*
LLH	Jumlah *low level heuristic* yang digunakan	-
T_0	Suhu awal algoritma Simulated Annealing	*Simulated Annealing*
T_1	Suhu akhir algoritma Simulated Annealing	*Simulated Annealing*
α	Penurunan suhu	*Simulated Annealing*
Nα	Jumlah iterasi setiap penurunan suhu	*Simulated Annealing*
β	Kenaikan suhu pada proses reheating	*Simulated Annealing*
Nβ	Jumlah iterasi setiap kenaikan suhu (*reheating*)	*Simulated Annealing*
TL	Panjang *tabu list* solusi algoritma	*Tabu Search*

Parameter	Keterangan	Algoritma
	Tabu Search	

Tabel 5.12 Skenario Uji Coba

Skenario	Algoritma	Iterasi (Juta)	Time Limit	NL / LLH	T_0	T_1	α	Nα	β	Nβ	TL
A	Self-Adaptive – Simulated Annealing	-	15 menit	8 / 4	95	0	0,999	50	0,5	25000	3
B	Self-Adaptive – Tabu Search – Simulated Annealing	-	15 menit	8 / 4	95	0	0,999	50	0,5	25000	3
C	Tabu Search – Simulated Annealing	-	15 menit	8 / 4	95	0	0,999	50	0,5	25000	3
D	Self-Adaptive – Tabu Search – Simulated Annealing (Pindah ruangan hanya pada LLH 1 Move)	-	15 menit	8 / 4	95	0	0,999	50	0,5	25000	3

5.5. Hasil Eksperimen

Pengujian terhadap satu instance dataset ITC2007 dan Socha dilakukan untuk melihat hubungan antara iterasi dan nilai fungsi obyektif. Sampel dari dataset ITC2007 adalah Hidden17, sedangkan sampel dari dataset Socha adalah Small3. Pembandingan akan dilakukan dengan algoritma Self Adaptif Simulated Annealing dan Tabu Search Simulated Annealing. Setelah pengujian sampel dilakukan, pengujian berdasarkan skenario uji coba akan dilakukan. Pada tabel 5.13. dapat dilihat keterangan judul yang terdapat pada tabel hasil eksperimen skenario dataset socha

Tabel 5.13 Keterangan Tabel Hasil Eksperimen Skenario Dataset Socha

Kode	Keterangan
Instance	Nama Instance yang diuji coba dalam hasil eksperimen
Avg Initial Sol	Rata-rata solusi awal. Solusi awal didapat ketika awal proses optimasi untuk pembentukan nilai solusi penjadwalan mata kuliah
Time	Waktu yang digunakan dalam proses optimasi penjadwalan
Best	Nilai penalti / solusi akhir yang paling baik yang didapatkan
Worst	Nilai penalti / solusi akhir yang paling buruk yang didapatkan
Avg	Nilai rata-rata penalti / solusi akhir yang didapatkan
St. dev	Standar deviasi dari nilai penalti / solusi akhir yang didapatkan

Pada tabel 5.14. dapat dilihat keterangan judul yang terdapat pada tabel hasil eksperimen skenario dataset itc2007. Tabel yang ditampilkan sedikit berbeda dengan tabel hasil dataset Socha, dikarenakan pada dataset itc2007 memiliki konsep *Distance to Feasibility*, *Feasibility*, dan tidak ada kolom *Time*. Tidak adanya kolom *Time* dikarenakan tidak ada solusi optimum yang tercapai pada dataset ini sehingga waktu yang dihabiskan sesuai dengan skenario uji coba hasil eksperimen pada tabel 5.12.

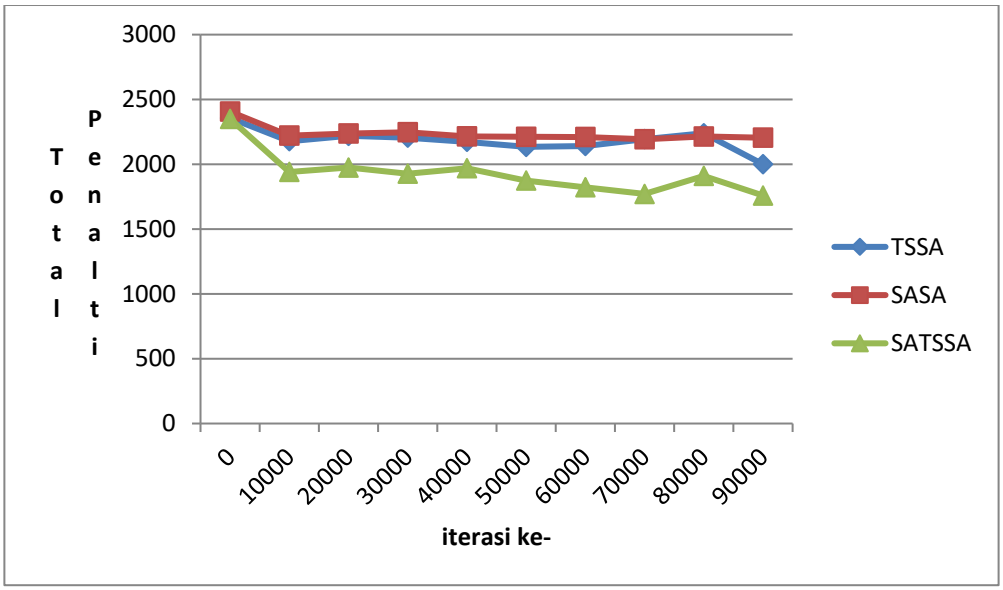
Tabel 5.14 Keterangan Tabel Hasil Eksperimen Skenario Dataset ITC2007

Kode	Keterangan

Kode	Keterangan
Instance	Nama Instance yang diuji coba dalam Algoritma *Self-Adaptive-Tabu-Simulated Annealing*
Avg Initial DTF	Rata-rata *Distance to Feasibility* (DTF) awal. Distance to feasibility adalah event yang tidak terjadwal agar sebuah event tidak menyebabkan solusi tidak layak karena melanggar *hard constraints*. DTF adalah total siswa yang mendaftar pada event yang tidak dijadwalkan tersebut. Detail DTF dibahas pada subbab 4.1.9.
Avg Initial Sol	Rata-rata solusi awal. Solusi awal didapat ketika awal proses optimasi untuk pembentukan nilai solusi penjadwalan mata kuliah
Feasibility	Prosentase feasibility. Pada dataset ITC2007, dikarenakan terdapat DTF maka **nilai penalti / solusi yang paling baik** adalah yg memiliki **nilai DTF paling kecil** atau **prosentase feasibility paling besar**. DTF dengan nilai kecil atau prosentase feasibility yang besar menandakan bahwa event yang terjadwal lebih banyak daripada yang tidak terjadwal.
Best	Nilai penalti / solusi akhir yang didapatkan bersamaan dengan nilai DTF yang paling kecil atau prosentase feasibility paling besar
Worst	Nilai penalti / solusi akhir yang didapatkan bersamaan dengan nilai DTF yang paling besar atau prosentase feasibility paling kecil
Avg Final DTF	Rata-rata *Distance to Feasibility* (DTF) akhir
Avg	Nilai rata-rata penalti / solusi akhir yang didapatkan
St. dev	Standar deviasi dari nilai penalti / solusi akhir yang didapatkan

5.5.1. Sampel Eksperimen

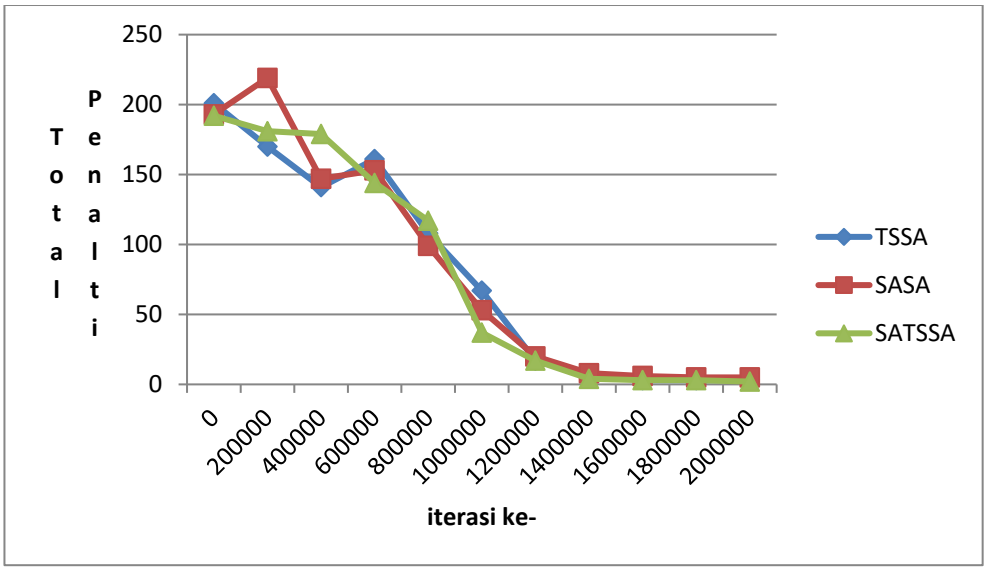
Pada gambar 5.3 dapat dilihat kinerja ketiga algoritma pada Sampel ITC Hidden17. Pada sampel eksperimen, kinerja Self Adaptive Tabu Search Simulated Annealing terlihat lebih baik dibandingkan kedua algoritma lainnya. Nilai penalti yang lebih baik adalah nilai penalti yang paling sedikit dengan grafik menurun.



Gambar 5.3 Sampel ITC Hidden17

Pada gambar 5.4 dapat dilihat kinerja ketiga algoritma pada Sampel Socha Small3. Pada sampel eksperimen, kinerja Self Adaptive Tabu Search Simulated Annealing juga terlihat lebih baik dibandingkan kedua algoritma lainnya.

Dari kedua sampel eksperimen tersebut menunjukkan karakteristik umum masing-masing algoritma dan bagaimana bila dikombinasikan. Algoritma *Simulated Annealing* dapat menerima solusi yang lebih jelek sehingga dapat menghindari terjebak di lokal optima. Algoritma *tabu search* dapat menghindari solusi yang terdapat pada *tabu list*, sehingga solusi tidak berulang. *Self-Adaptive* dapat menyeleksi dan tetap memilih kembali LLH yang memberikan solusi yang lebih baik, sehingga solusi bisa diperbaiki dengan lebih baik.



Gambar 5.4 Sampel Socha Small3

5.5.2. Hasil Skenario A

Pada tabel 5.15. dapat dilihat hasil skenario A pada dataset ITC 2007. Feasibility terbaik berkisar antara 80% hingga 100%. Walaupun solusi memiliki DTF, namun solusi masih valid / layak dikarenakan solusi tidak melanggar *hard constraints*. *Distance to Feasibility* (DTF) dari akhir dapat lebih baik yaitu dengan nilai lebih kecil daripada DTF di awal. Begitu juga dengan nilai akhir *objective function* atau nilai penalti, dapat lebih kecil dari pada nilai di awal. Keterangan mengenai judul tabel hasil eksperimen skenario ITC2007 dapat dilihat pada tabel 5.14.

Tabel 5.15. Hasil Skenario A Dataset ITC2007

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
1	1655	1951,33	84%	1982	81% - 1957	1651	1943	47,6
2	1934	1902	81%	1806	78% - 1820	1911,33	1890	133,5
3	736,67	3209,67	97%	2306	95% - 2538	593	2496,67	173,7

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
4	1508,67	2975	89%	2766	87% - 2577	1457,33	2720,33	126,8
5	847	1131	88%	1166	85% - 1135	814,33	1148,33	15,9
6	913,33	1152,67	89%	1202	85% - 1176	853,67	1173	30,6
7	903	1575	88%	1530	1515	827	1487,67	60,8
8	645,33	1699,67	91%	1762	89% - 1615	640,67	1681,33	74,5
9	1532,67	1934	84%	2023	1901	1520	1945,67	67,2
10	2053,67	1814	80%	1733	1981	2048,33	1815,67	143,2
11	1345	2952	93%	2357	90% - 2697	1243	2542,67	172,1
12	2008	2941,33	86%	2870	84% - 2696	1891,33	2769,67	90,0
13	1081,67	1166,33	86%	1155	84% - 1140	1006,33	1160	22,9
14	955,33	1150	87%	1200	85% - 1041	944	1147,67	92,4
15	647,67	1501	91%	1426	90% - 1426	599	1428,67	4,6
16	246,33	1694,33	98%	1526	95% - 1587	184,33	1527,67	58,5
17	0	2110,67	100%	1098	1229	0	1178	70,1
18	1602,33	2058	85%	2000	83% - 2004	1442,67	2001,33	2,3
19	2082	2871,67	86%	2820	83% - 2696	1998,33	2711	102,3
20	489,33	2951	97%	2990	95% - 2954	489,33	2951	40,6
21	590,33	1217	92%	1259	90% - 1242	566,33	1225,67	43,8

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
22	2394,33	1651,67	80%	1688	78% - 1734	2274,33	1683,33	53,1
23	4683,33	3451	80%	3674	77% - 3354	4574,67	3453,33	191,4
24	1695	2964,67	90%	2950	88% - 2909	1581,33	2968,67	70,9

Pada tabel 5.16 dapat dilihat hasil skenario A pada dataset Socha. Pada dataset ini semua mata kuliah dapat terjadwal. Pada dataset kategori *small*, dapat mencapai nilai optimum 0. Nilai akhir *objective function* atau nilai penalti dapat lebih kecil dari pada nilai di awal. Keterangan mengenai tabel hasil eksperimen skenario socha dapat dilihat pada tabel 5.13.

Tabel 5.16 Hasil Skenario A Dataset Socha

Instance	Avg Initial Sol	Time	Best	Worst	Avg	St.dev
small1	198	53s	0	1	0,33	0,6
small2	226,33	2m59s	0	1	0,67	0,6
small3	201,33	3m45s	0	3	1,33	1,5
small4	121,67	14m4s	0	3	1,67	1,5
small5	236,67	47s	0	0	0	0
medium1	792,67	15m	207	225	213,67	9,9
medium2	809,67	15m	205	218	213	7
medium3	809,67	15m	244	259	253,67	8,4
medium4	815,33	15m	202	222	210	10,6
medium5	734,33	15m	168	235	192	37,3
Large	1540,33	15m	920	1026	984	56,3

5.5.3. Hasil Skenario B

Pada tabel 5.17. dapat dilihat hasil skenario B pada dataset Socha. Pada dataset ini semua mata kuliah dapat terjadwal. Pada dataset kategori *small*, dapat mencapai nilai optimum 0. Nilai akhir *objective function* atau nilai penalti dapat

lebih kecil dari pada nilai di awal. Keterangan mengenai tabel hasil eksperimen skenario socha dapat dilihat pada tabel 5.13.

Tabel 5.17 Hasil Skenario B Dataset Socha

Instance	Avg Initial Sol	Time	Best	Worst	Avg	St.dev
small1	195,67	5m17s	0	2	1	1
small2	218,67	59s	0	2	0,67	1,1
small3	193,33	14m38s	0	1	0,67	0,6
small4	131,67	7m	0	2	1	1
small5	241,33	4m11s	0	4	2	2
medium1	789,33	15m	197	225	208,33	14,7
medium2	814	15m	199	238	213,67	21,2
medium3	810,33	15m	254	262	258,33	4
medium4	824	15m	177	252	216,33	37,6
medium5	807	15m	148	227	177,67	43
Large	1591	15m	971	1122	1038	76,9

Pada tabel 5.18. dapat dilihat hasil skenario B pada dataset ITC 2007. Feasibility terbaik berkisar antara 79% hingga 100%. Walaupun solusi memiliki DTF, namun solusi masih valid / layak dikarenakan solusi tidak melanggar *hard constraints*. *Distance to Feasibility* (DTF) dari akhir dapat lebih baik yaitu dengan nilai lebih kecil daripada DTF di awal. Begitu juga dengan nilai akhir *objective function* atau nilai penalti, dapat lebih kecil dari pada nilai di awal. Keterangan mengenai tabel hasil eksperimen skenario ITC2007 dapat dilihat pada tabel 5.14.

Tabel 5.18 Hasil Skenario B Dataset ITC2007

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
1	1570	1960,67	85%	2012	83% - 1949	1556	1947,33	65,5
2	1796	1833	83%	1879	81% - 1760	1746	1835	65,2

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
3	765,67	3465,67	96%	2158	95% - 2730	673,33	2733,67	577,5
4	1414,33	3027,67	90%	3013	89% - 3089	1305	2948,67	181,3
5	885,67	1143,33	86%	1181	85% - 1134	870	1160	23,9
6	957,33	1103,67	87%	1085	85% - 1070	920	1109	55,1
7	818,67	1444,67	89%	1377	87% - 1452	796,67	1380,33	70,1
8	720	1612,67	90%	1557	1608	649	1545,33	69,2
9	1504	2019,67	86%	2061	84% - 2021	1500,67	2025	34,2
10	2118,33	1746,33	80%	1766	77% - 1763	2100, 67	1755, 67	15,4
11	1459,33	2954,67	91%	2286	2346	1287,67	2298,67	42,4
12	1787,33	2957,33	88%	2837	85% - 3096	1662,33	2872,33	208,3
13	850,67	1183,67	90%	1207	86% - 1158	798	1182,67	24,5
14	1062,67	1128,33	86%	1138	82% - 1136	1038	1126,33	18,5
15	650,67	1478,67	93%	1392	88% - 1511	582,67	1471	68,4
16	281	1704,33	98%	1516	95% - 1456	188	1485,67	30
17	0	2260,33	100%	1135	1488	0	1279,67	184,9
18	1566	1880	85%	1976	83% - 1623	1404	1875,33	220
19	2069,33	2931,67	86%	2920	83% - 2601	2025	2793	169,1
20	390	3075	97%	3207	96% -	390	3075	133,5

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
					2940			
21	673	1178	90%	1182	89% - 1163	650	1182,67	20
22	2156,33	1782,67	81%	1806	79% - 1846	2113	1812	31,4
23	4777,67	3400	79%	3275	77% - 3611	4721,33	3429,67	169,5
24	1838	2954	88%	2988	86% - 2864	1808,33	2956,33	81,3

5.5.4. Hasil Skenario C

Pada tabel 5.19. dapat dilihat hasil skenario C pada dataset ITC2007. *Feasibility* terbaik berkisar antara 80% hingga 100%. Walaupun solusi memiliki DTF dan *feasibility* tidak mencapai 100%, namun solusi masih dikatakan layak karena tidak melanggar *hard constraints*. Hasil yang didapatkan dari scenario C yaitu algoritma tidak mampu memperbaiki *feasibility* terlihat dari kolom avg initial DTF dan avg final DTF yang memiliki nilai sama pada semua *instance*. Namun algoritma mampu mengoptimasi nilai penalti *soft constraints* terlihat dari kolom avg initial sol dan avg. Keterangan mengenai tabel hasil eksperimen skenario ITC2007 dapat dilihat pada tabel 5.14.

Tabel 5.19 Hasil Skenario C Dataset ITC2007

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
1	1584,67	1927,67	84%	1868	83% - 1963	1584,67	1930,33	54
2	1961	1806,33	80%	1789	79% - 1805	1961	1803,33	13,6
3	902,67	3332,67	94%	2848	93% - 2926	902,67	2836,33	96

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
4	1379,33	3184	91%	3145	87% - 3364	1379,33	3184	164
5	906,33	1161	86%	1208	85% - 1142	906,33	1161	41
6	888	1122,33	88%	1112	85% - 1086	888	1122,33	42,45
7	850,33	1435,33	90%	1441	86% - 1452	850,33	1435,33	20,1
8	655	1564,67	91%	1553	89% - 1610	655	1564,67	40,8
9	1316,33	2030	87%	2098	85% - 1980	1316,33	2030	61
10	2079,33	1748,33	80%	1717	78% - 1789	2079,33	1748,33	36,9
11	1383,67	2994,67	91%	2866	89% - 3080	1383,67	2994,67	113,4
12	1910	3086,67	86%	3109	83% - 3237	1910	3086,67	162,6
13	997	1177	86%	1156	84% - 1125	997	1177	65,1
14	992,33	1103,67	87%	1142	83% - 999	992,33	1103,67	91,7
15	708,67	1493	89%	1432	1542	708,67	1493	56
16	325,67	1677,67	95%	1613	93% - 1767	325,67	1677,67	79,9
17	0	2289	100%	1003	1305	0	1173	154,5
18	1596	1898,67	84%	1985	80% - 1821	1596	1898,67	82,3
19	1963,67	2965,33	86%	3003	85% - 2939	1963,67	2965,33	33,5
20	417,67	2997,67	97%	2918	96% - 3029	417,67	2997,67	69,5

Instance	Avg Initial DTF	Avg Initial Sol	Feasibility	Best	Worst	Avg Final DTF	Avg	St.dev
21	595,33	1198,67	92%	1199	89% - 1165	595,33	1198,67	33,5
22	2314,67	1696	80%	1808	76% - 1619	2314,67	1696	99,2
23	4466	3636,67	80%	3819	78% - 3586	4466	3636,67	163
24	1856,67	2897,33	88%	2871	86% - 2830	1856,67	2897,33	83,7

Pada tabel 5.20. dapat dilihat hasil skenario C pada dataset Socha. Pada dataset ini semua mata kuliah dapat terjadwal. Pada dataset kategori *small*, dapat mencapai nilai optimum 0. Nilai akhir *objective function* atau nilai penalti dapat lebih kecil dari pada nilai di awal. Keterangan mengenai tabel hasil eksperimen skenario socha dapat dilihat pada tabel 5.13.

Tabel 5.20 Hasil Skenario C Dataset Socha

Instance	Avg Initial Sol	Time	Best	Worst	Avg	St.dev
small1	203,33	45s	0	1	0,33	0,6
small2	225,33	1m47s	0	4	2	2
small3	190	6m16s	0	1	0,33	0.6
small4	130,33	2m40s	0	1	0,33	0,6
small5	247,67	1m	0	0	0	0
medium1	777,67	15m	210	240	220,67	16,8
medium2	779,67	15m	209	222	217,67	7,5
medium3	806,33	15m	232	251	244,67	11
medium4	815,67	15m	177	230	200,33	27,1
medium5	826,33	15m	139	181	163	21,6
Large	1572	15m	864	1142	987,67	141,5

5.5.5. Hasil Skenario D

Pada tabel 5.21. dapat dilihat hasil skenario D pada dataset ITC2007. *Feasibility* terbaik berkisar antara 80% hingga 100%. Walaupun solusi memiliki DTF dan *feasibility* tidak mencapai 100%, namun solusi masih dikatakan layak karena tidak melanggar *hard constraints*. Hasil yang didapatkan dari scenario yaitu algoritma mampu memperbaiki *feasibility* terlihat dari kolom avg initial DTF dan avg final DTF yang memiliki nilai yang semakin berkurang pada *instance* 3, 4, 5, 7, 12, 13, 14, 18, 22, dan 24. Selain itu, algoritma mampu mengoptimasi nilai penalti *soft constraints* terlihat dari kolom avg initial sol dan avg. Keterangan mengenai tabel hasil eksperimen skenario ITC2007 dapat dilihat pada tabel 5.14.

Tabel 5.21 Hasil Skenario D Dataset ITC2007

Instance	Avg Initial Distance to Feasibility	Avg Initial Sol	Prosentase Feasibility	Best	Worst	Avg Final Distance to Feasibility	Avg	St.dev
1	1616,33	1920,67	84%	1765	83% - 1703	1616,33	1769,67	69,1
2	1868,67	1905,33	81%	1749	80% - 1756	1868,67	1777	42,6
3	966,67	3311	97%	1954	93% - 1835	764,33	1835,33	118,5
4	1423	3021	91%	2076	86% - 2127	1389	2033,33	120,8
5	863,33	1111,67	87%	1143	86% - 1083	826,67	1105,33	32,8
6	881,67	1134,67	87%	1125	86% - 1133	881,67	1112	29,7
7	1004	1546,33	87%	1329	85% - 1156	940,67	1237	87
8	670,67	1566	92%	1125	87% - 1125	670,67	1174	84,9
9	1494	2013,67	85%	1980	84% - 1916	1494	1892	102,1
10	1972	1793,67	80%	1882	78% - 1710	1972	1782,67	89
11	1361,33	2861,67	92%	2146	89% - 2086	1361,33	2117,67	30,1
12	1765,67	2915	88%	2055	86% - 2359	1726	2300	221,5
13	1019,33	1161	87%	1191	84% - 1100	982	1120,67	62,6
14	1051,67	1124,67	85%	1118	84% -	1000,33	1069,33	45,8

Instance	Avg Initial Distance to Feasibility	Avg Initial Sol	Prosentase Feasibility	Best	Worst	Avg Final Distance to Feasibility	Avg	St.dev
					1063			
15	513	1590,33	93%	1264	91% - 1331	513	1292,33	34,7
16	264,33	1585,67	96%	890	95% - 1100	264,33	991	105,2
17	0	2360	100%	412	80,53	0	503,67	80,5
18	1456,67	1963,33	88%	1890	81% - 1785	1146,33	1795	90,4
19	2180,33	3044,67	84%	2412	83% - 2449	2180,33	2406,67	45,2
20	415,33	3053,33	97%	2395	96% - 2459	415,33	2427,33	32
21	632,33	1198	91%	1136	89% - 1116	632,33	1123	11,3
22	2258,67	1733,33	81%	1762	79% - 1696	2199,33	1722,67	34,8
23	4611,67	3373,67	80%	3287	77% - 3080	4611,67	3133,33	135,1
24	1700,67	2991,33	90%	2673	86% - 2293	1693	2398,33	240

5.6.Perbandingan Hasil Eksperimen

Dari keempat skenario algoritma, dibandingkan pada masing-masing dataset.

5.6.1. Perbandingan Hasil Eksperimen Dataset Socha

Pada tabel 5.22. dapat dilihat keterangan judul tabel perbandingan hasil eksperimen skenario dataset Socha.

Tabel 5.22 Keterangan Tabel Perbandingan Hasil Eksperimen Skenario Dataset Socha

Kode	Keterangan	Skenario Uji Coba
SASA	*Self Adaptive - Simulated Annealing*	A
SATSSA	*Self Adaptive - Tabu Search - Simulated Annealing*	B
TSSA	*Tabu Search - Simulated Annealing*	C
Instance	Nama Instance yang diuji coba dalam hasil eksperimen	-
Time	Waktu yang digunakan dalam proses optimasi penjadwalan	-
Best	Nilai penalti / solusi akhir yang paling baik yang didapatkan	-
Avg	Nilai rata-rata penalti / solusi akhir yang didapatkan	-

Dapat dilihat pada tabel 5.23 perbandingan hasil eksperimen dataset Socha. Ketiga algoritma mampu mencapai hasil optimum pada kategori *small*.

Pada medium1 dan medium2, algoritma SATSSA mencapai solusi yang lebih kecil yaitu 197 dan 199. Pada medium3, medium5, dan large, algoritma TSSA dapat mencapai solusi yang lebih kecil yaitu 232, 139, dan 864. Sedangkan pada medium 4, algoritma SATSSA dan algoritma TSSA sama-sama mencapai solusi yang paling kecil yaitu 177. Pada solusi dengan nilai paling optimum, dapat dilihat waktu yang diperlukan dalam proses optimasi. Algoritma SATSSA memerlukan waktu yang paling sedikit pada *instance small2* yakni 59 detik.

Tabel 5.23 Perbandingan Hasil Eksperimen Dataset Socha

Instance	**SASA**			**SATSSA**			**TSSA**		
	Time	**Best**	**Avg**	**Time**	**Best**	**Avg**	**Time**	**Best**	**Avg**
small1	53s	**0**	0,33	5m17s	**0**	1	**45s**	**0**	**0,33**
small2	2m59s	**0**	**0,67**	**59s**	**0**	**0,67**	1m47s	**0**	2
small3	**3m45s**	**0**	1,33	14m38s	**0**	0,67	6m16s	**0**	**0,33**
small4	14m4s	**0**	1,67	7m	**0**	1	**2m40s**	**0**	**0,33**
small5	**47s**	**0**	**0**	4m11s	**0**	2	1m	**0**	**0**
medium1	15m	207	213,67	15m	**197**	**208,33**	15m	210	220,67
medium2	15m	205	**213**	15m	**199**	213,67	15m	209	217,67
medium3	15m	244	253,67	15m	254	258,33	15m	**232**	**244,67**
medium4	15m	202	210	15m	**177**	216,33	15m	**177**	**200,33**
medium5	15m	168	192	15m	148	177,67	15m	**139**	**163**
Large	15m	920	**984**	15m	971	1038	15m	**864**	987,67

5.6.2. Perbandingan Hasil Eksperimen Dataset ITC 2007

Dapat dilihat pada tabel 5.24. keterangan tabel perbandingan hasil eksperimen dataset ITC2007.

Tabel 5.24 Keterangan Tabel Perbandingan Hasil Eksperimen Skenario Dataset ITC2007

Kode	**Keterangan**	**Skenario Uji Coba**
SASA	*Self Adaptive - Simulated Annealing*	A
SATSSA	*Self Adaptive - Tabu Search - Simulated Annealing*	B
TSSA	*Tabu Search - Simulated Annealing*	C
SATSSA (Room move 1)	*Self Adaptive - Tabu Search - Simulated Annealing* dengan pemindahan ruangan hanya pada LLH move1 TS	D
Instance	Nama Instance yang diuji coba dalam hasil eksperimen	-
Feas	Prosentase feasibility. Pada dataset ITC2007, dikarenakan terdapat DTF maka **nilai penalti / solusi yang paling baik** adalah yg memiliki **nilai DTF**	-

	paling kecil atau **prosentase feasibility paling besar**. DTF dengan nilai kecil atau prosentase feasibility yang besar menandakan bahwa event yang terjadwal lebih banyak daripada yang tidak terjadwal.	
Best	Nilai penalti / solusi akhir yang didapatkan bersamaan dengan nilai DTF yang paling kecil atau prosentase feasibility paling besar	-

Tabel 5.25 Perbandingan Hasil Eksperimen Dataset ITC2007

Instance	**SASA**		**SATSSA**		**TSSA**		**SATSSA (Room move 1)**	
	Feas	**Best**	**Feas**	**Best**	**Feas**	**Best**	**Feas**	**Best**
1	84%	1982	**85%**	**2012**	84%	1868	84%	1765
2	81%	1806	**83%**	**1879**	80%	1789	81%	1749
3	97%	2306	96%	2158	94%	2848	**97%**	**1954**
4	89%	2766	90%	3013	**91%**	**3145**	91%	2076
5	**88%**	**1166**	86%	1181	86%	1208	87%	1143
6	**89%**	**1202**	87%	1085	88%	1112	87%	1125
7	88%	1530	89%	1377	**90%**	**1441**	87%	1329
8	91%	1762	90%	1557	**91%**	**1553**	92%	1125
9	84%	2023	86%	2061	**87%**	**2098**	85%	1980
10	**80%**	**1733**	80%	1766	80%	1717	80%	1882
11	**93%**	**2357**	91%	2286	91%	2866	92%	2146
12	86%	2870	88%	2837	86%	3109	**88%**	**2055**
13	86%	1155	**90%**	**1207**	86%	1156	87%	1191
14	**87%**	**1200**	86%	1138	87%	1142	85%	1118
15	91%	1426	93%	1392	89%	1432	93%	1264
16	98%	1526	**98%**	**1516**	95%	1613	**96%**	**890**
17	100%	1098	100%	1135	100%	1003	**100%**	**412**
18	85%	2000	85%	1976	84%	1985	**88%**	**1890**
19	86%	2820	**86%**	**2920**	86%	3003	**84%**	**2412**
20	97%	2990	97%	3207	97%	2918	97%	2395
21	**92%**	**1259**	90%	1182	92%	1199	**91%**	**1136**
22	80%	1688	**81%**	**1806**	80%	1808	**81%**	**1762**
23	80%	3674	79%	3275	**80%**	**3819**	**80%**	**3287**
24	**90%**	**2950**	88%	2988	88%	2871	**90%**	**2673**

Pada perbandingan hasil eksperimen datset ITC 2007 pada tabel 5.25 solusi yang lebih baik adalah prosentase feasibility yang lebih besar atau nilai DTF yang lebih kecil. Apabila nilai DTF sama, maka nilai penalti yang lebih kecil merupakan solusi yang lebih baik. Dari ketiga algoritma, *Self-Adaptive Tabu Search Simulated Annealing* secara kumulatif menghasilkan nilai solusi yang paling baik pada 12 instances yaitu pada instance 1, 2, 3, 12, 13, 15, 16, 17, 18, 19, 20, dan 22. Algoritma Self Adaptive Simulated Annealing memiliki 7 instances dengan solusi yang lebih baik, sedangkan algoritma Tabu Search Simulated Annealing hanya 5 instances. Pada algoritma dengan feasibility 100% pada instances 17, algoritma SATSSA mampu memberikan solusi yang lebih optimal.

5.7.Perbandingan Benchmark

Perbandingan benchmark perlu dilakukan untuk mengukur performa algoritma dengan penelitian terdahulu. Perbandingan benchmark dilakukan pada kedua dataset yaitu dataset socha dan dataset ITC2007.

5.7.1. Perbandingan Benchmark Socha

Perbandingan benchmark socha dilakukan dengan penelitian terdahulu menggunakan dataset socha pada domain yang sama yaitu Post Enrollment Course Timetabling Problem. Daftar algoritma Benchmark Socha yang digunakan dapat dilihat pada tabel 5.26.

Tabel 5.26 Daftar Algoritma Benchmark Socha

Kode	**Algoritma**	**Peneliti**
SATSSA	Self Adaptive – Tabu Search – Simulated Annealing	Algoritma yang diusulkan dalam penelitian ini
TSA	Tabu Search Simulated Annealing [17]	Ahsanul
TVNS	Tabu - Variable Neighbourhood Search [18]	Salwani Abdullah, dkk.
MBO	Migrating Bird Optimization [19]	Hishammudin Asmuni, dkk.
FMH	Fuzzy Multiple Heuristics [20]	Hishammudin Asmuni, dkk.
GC	Graph Coloring [21]	Edmund K. Burke, dkk.

Pada perbandingan benchmark Socha tabel 5.27 dapat dilihat perbandingan dengan algoritma yang telah diusulkan. Algoritma SATSSA

menghasilkan nilai yg lebih optimum dibandingkan dengan algoritma lain pada instance medium 1 dan medium 4. Pada instance small1, small2, small3, dan small5 algoritma SATSSA menghasilkan nilai yang optimum sama dengan algoritma TSA, dan MBO.

Tabel 5.27 Perbandingan Benchmark Socha

Instance	**SATSSA**		**TSA**	**TVNS**	**MBO**	**FMH**	**GC**
	Best	**Avg**	**Best**	**Best**	**Best**	**Best**	**Best**
small1	**0**	1	**0**	25	**0**	10	6
small2	**0**	0,67	**0**	22	**0**	9	7
small3	**0**	0,67	**0**	19	**0**	7	3
small4	**0**	1	**0**	14	**0**	17	3
small5	**0**	2	**0**	17	**0**	7	4
medium1	**197**	208,33	317	394	198	243	372
medium2	199	213,67	313	378	**195**	325	419
medium3	254	258,33	357	305	**208**	249	359
medium4	**177**	216,33	247	282	181	285	348
medium5	148	177,67	292	276	**116**	132	171
Large	971	1038	**932**	1015	936	1138	1068

5.7.2. Perbandingan Benchmark ITC2007

Perbandingan benchmark socha dilakukan dengan penelitian terdahulu menggunakan dataset socha pada domain yang sama yaitu Post Enrollment Course Timetabling Problem. Daftar algoritma Benchmark Socha yang digunakan dapat dilihat pada tabel 5.28. keterangan tabel perbandingan dataset ITC2007 dapat dilihat pada tabel 5.24.

Tabel 5.28 Daftar Algoritma Benchmark ITC2007

Kode	**Algoritma**	**Peneliti**
SATSSA	Self Adaptive – Tabu Search – Simulated Annealing	Algoritma yang diusulkan dalam penelitian ini
CIHLS	Constructive + iterated heuristic + local search(SA) [22]	Lewis
ACO	Ant Colony Optimisation [23]	Mayer, dkk

Pada perbandingan benchmark ITC2007 tabel 5.29 dapat dilihat perbandingan dengan algoritma yang telah diusulkan. Algoritma SATSSA menghasilkan nilai yg lebih optimum dibandingkan dengan algoritma lain pada instance 1, 2, 20, 21, dan 23.

Tabel 5.29 Perbandingan Benchmark ITC 2007

Instance	SATSSA		CIHLS		ACO	
	%Feas	Best	%Feas	Best	%Feas	Best
1	**84**	**1765**	45	1294	54	0
2	**81**	**1749**	22	1599	59	0
3	97	1954	>95	278	**100**	**110**
4	91	2076	>95	388	**100**	**53**
5	87	1143	>95	22	**100**	**13**
6	87	1125	**>95**	**369**	95	0
7	87	1329	>95	74	**100**	**0**
8	92	1125	100	0	**100**	**0**
9	85	1980	2	1482	**85**	**0**
10	80	1882	2	2380	**100**	**0**
11	92	2146	>95	344	**99**	**143**
12	88	2055	**>95**	**486**	86	0
13	87	1191	**>95**	**365**	94	5
14	85	1118	>95	222	**100**	**0**
15	93	1264	>95	266	**100**	**0**
16	96	890	>95	99	**100**	**0**
17	100	412	-	-	**100**	**68**
18	88	1890	-	-	**100**	**26**
19	84	2412	-	-	**90**	**22**
20	**97**	**2395**	-	-	0	-
21	**91**	**1136**	-	-	80	33
22	81	1762	-	-	**100**	**0**
23	**80**	**3287**	-	-	0	-
24	90	2673	-	-	**100**	**30**

Hasil yang didapatkan dari algoritma SATSSA belum mencapai hasil yang optimum jika dibandingkan dengan algoritma CIHLS dan ACO dikarenakan belum ada instance yang mencapai feasibility 100% dengan nilai penalti 0. Pada algoritma CIHLS, memiliki tahapan optimasi yang dibagi dalam 3 tahap. Tahapan pertama, menjadwalkan sebanyak mungkin mata kuliah yang bisa terjadwal dan memenuhi Hard Constraints 1 hingga 4. Tahapan kedua, mata kuliah yang telah terjadwal diusahakan dapat memenuhi Hard Constraints 5 tanpa melanggar Hard Constraints 1 hingga 4. Kemudian tahapan akhir, optimasi nilai penalti Soft

Constraints tanpa melanggar Hard Constraints 1 hingga 5. Selain itu, Heuristic yang digunakan pada algoritma CIHLS berjumlah 7.

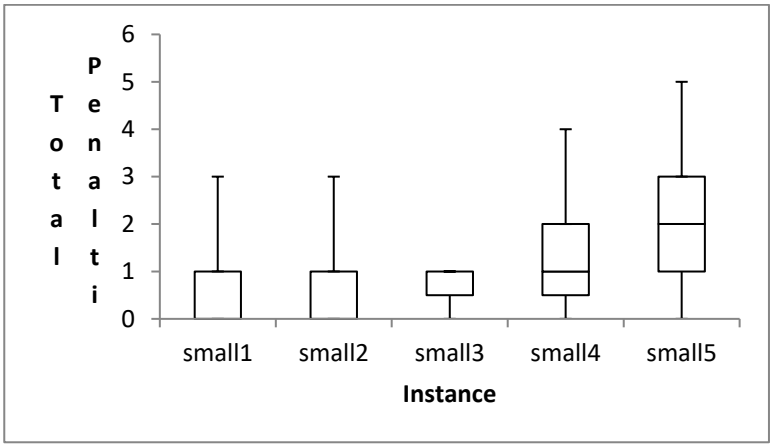
Algoritma ACO menggunakan dua matrix pheromone yang berbeda dan dikombinasikan dengan prosedur yang efektif sehingga dapat membangun solusi yang lebih baik. Langkah-langkah pada ACO dalam membangun solusi adalah *artificial ants, pheromone update,* dan aksi daemon opsional seperti melakukan pencarian lokal.

5.8.Boxplot Hasil Eksperimen

Pada subbab boxplot hasil eksperimen akan dibahas boxplot hasil eksperimen dataset socha dan dataset ITC 2007.

5.8.1. Boxplot Hasil Eksperimen Dataset Socha

Pada subbab ini menampilkan boxplot untuk dataset Socha. Boxplot digunakan untuk membandingkan distribusi persentil dari algoritma SATSSA.



Gambar 5.5 Boxplot Hasil Eksperimen Dataset Socha Small

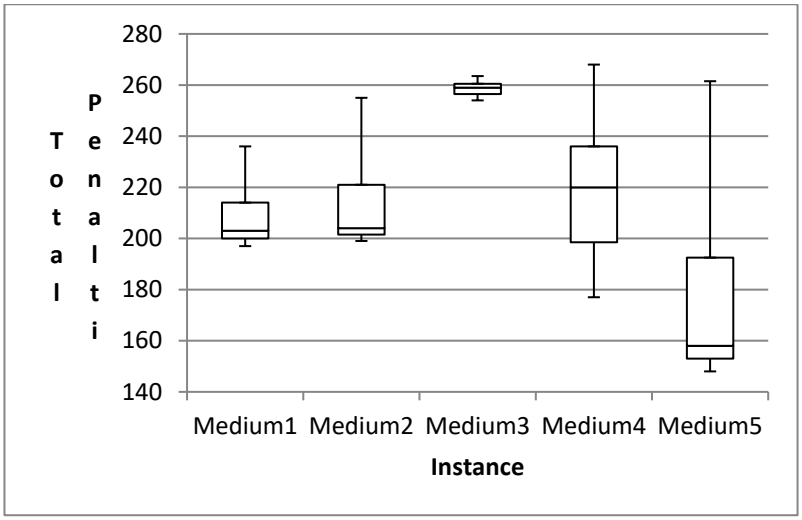
Pada gambar 5.5 menggambarkan solusi akhir yang didapat dari *running* program beberapa kali. Diagram boxplot tersebut menggambarkan kinerja dari algoritma SATSSA. Pada tabel 5.30 dapat dilihat perhitungan nilai minimal, Q1, median, Q3, dan nilai maksimum dari dataset Socha kategori small yang diacu.

Tabel 5.30 Perhitungan Boxplot Hasil Eksperimen Dataset Socha Small

	small1	**small2**	**small3**	**small4**	**small5**

	small1	small2	small3	small4	small5
Min	0	0	0	1	0
Q1	0.5	0	0.5	1.5	1
Med	1	0	1	2	2
Q3	1.5	1	1	3	3
Max	2	2	1	4	4

Pada gambar 5.6 menggambarkan solusi akhir yang didapat dari *running* program dataset Socha kategori Medium.



Gambar 5.6 Boxplot Hasil Eksperimen Dataset Socha Medium

Pada tabel 5.31 dapat dilihat perhitungan nilai minimal, Q1, median, Q3, dan nilai maksimum dari dataset Socha kategori medium yang diacu.

Tabel 5.31 Perhitungan Boxplot Hasil Eksperimen Dataset Socha Medium

	Medium1	Medium2	Medium3	Medium4	Medium5
Min	197	199	254	177	148
Q1	200	201.5	256.5	198.5	153
Med	203	204	259	220	158
Q3	214	221	260.5	236	192.5
Max	225	238	262	252	227

Pada gambar 5.7 menggambarkan solusi akhir yang didapat dari *running* program dataset Socha kategori Large.

Gambar 5.7 Boxplot Hasil Eksperimen Dataset Socha Large

Pada tabel 5.32 dapat dilihat perhitungan nilai minimal, Q1, median, Q3, dan nilai maksimum dari dataset Socha kategori large yang diacu.

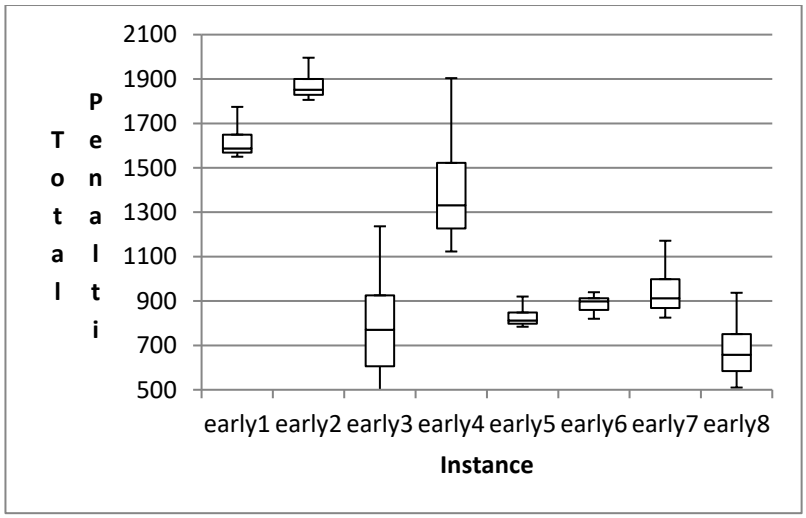
Tabel 5.32 Perhitungan Boxplot Hasil Eksperimen Dataset Socha Large

	Large
Min	971
Q1	996
Med	1021
Q3	1071.5
Max	1122

Secara keseluruhan, solusi yang dihasilkan menggunakan algoritma SATSSA pada dataset Socha lebih baik daripada hasil pada dataset ITC2007.

5.8.2. Boxplot Hasil Eksperimen Dataset ITC2007

Pada subbab ini menampilkan boxplot untuk dataset ITC2007. Boxplot digunakan untuk membandingkan distribusi persentil dari algoritma SATSSA. Pada gambar 5.8 menggambarkan solusi akhir yang didapat dari *running* program dataset ITC2007 kategori Early.



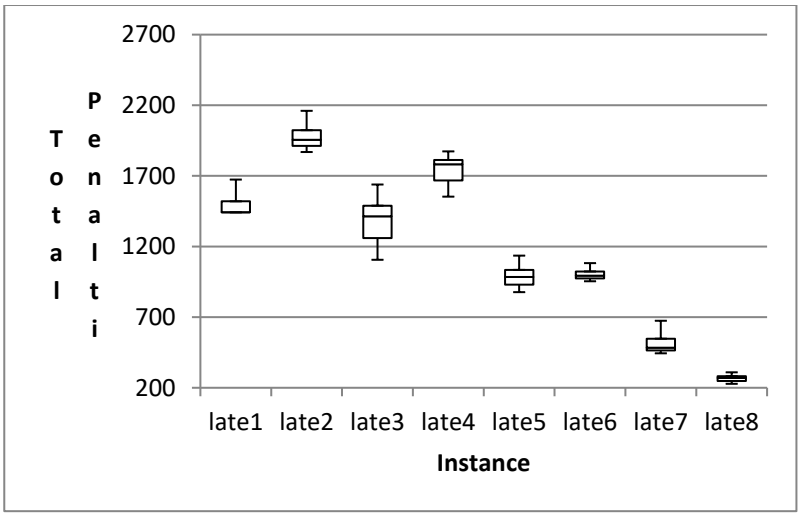
Gambar 5.8 Boxplot Hasil Eksperimen Dataset ITC2007 Early

Pada tabel 5.33 dapat dilihat perhitungan nilai minimal, Q1, median, Q3, dan nilai maksimum dari dataset ITC2007 kategori early yang diacu.

Tabel 5.33 Perhitungan Boxplot Hasil Eksperimen Dataset ITC2007 Early

	Early1	Early2	Early3	Early4	Early5	Early6	Early7	Early8
Min	1550	1806	442	1123	784	820	825	510
Q1	1568.5	1829	606	1227	798	859.5	868.5	584
Med	1587	1852	770	1331	812	899	912	658
Q3	1649.5	1900	925.5	1522	848	912.5	998.5	751
Max	1712	1948	1081	1713	884	926	1085	844

Pada gambar 5.9 menggambarkan solusi akhir yang didapat dari *running* program dataset ITC2007 kategori Late.



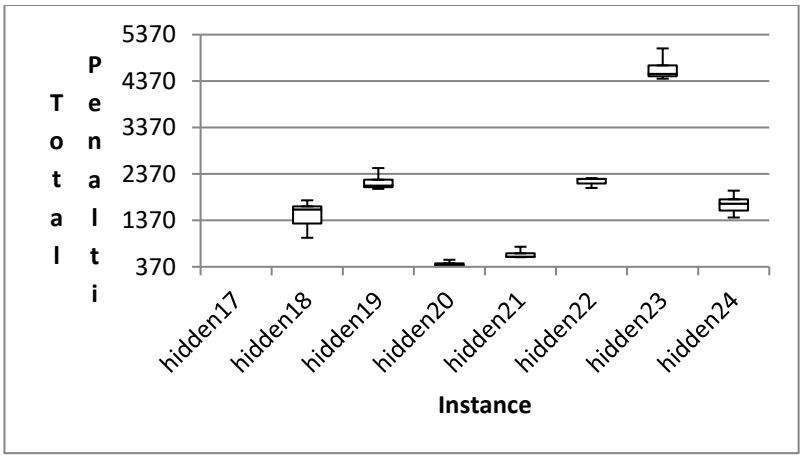
Gambar 5.9 Boxplot Hasil Eksperimen Dataset ITC2007 Late

Pada tabel 5.34 dapat dilihat perhitungan nilai minimal, Q1, median, Q3, dan nilai maksimum dari dataset ITC2007 kategori late yang diacu.

Tabel 5.34 Perhitungan Boxplot Hasil Eksperimen Dataset ITC2007 Late

	late1	late2	late3	late4	late5	late6	late7	late8
Min	1442	1869	1106	1553	877	954	444	228
Q1	1442.5	1912	1260	1667.5	930.5	974	464	248.5
Med	1443	1955	1414	1782	984	994	484	269
Q3	1520	2023.5	1489	1812.5	1034.5	1023.5	547.5	282.5
Max	1597	2092	1564	1843	1085	1053	611	296

Pada gambar 5.10 menggambarkan solusi akhir yang didapat dari *running* program dataset ITC2007 kategori Hidden.



Gambar 5.10 Boxplot Hasil Eksperimen Dataset ITC2007 Hidden

Pada tabel 5.35 dapat dilihat perhitungan nilai minimal, Q1, median, Q3, dan nilai maksimum dari dataset ITC2007 kategori hidden yang diacu.

Tabel 5.35 Perhitungan Boxplot Hasil Eksperimen Dataset ITC2007 Hidden

	hidden17	hidden18	hidden19	hidden20	hidden21	hidden22	hidden23	hidden24
Min	0	995	2049	354	577	2068	4421	1432
Q1	0	1301	2085	381	583	2163.5	4472.5	1581
Med	0	1607	2121	408	589	2259	4524	1730
Q3	0	1672	2246	446	660	2265	4707	1823.5
Max	0	1737	2371	484	731	2271	4890	1917

Halaman ini sengaja dikosongkan

BAB 6
KESIMPULAN DAN SARAN

6.1.Kesimpulan

Kombinasi antara Self-Adaptive, Tabu Search, dan Simulated Annealing melalui pendekatan *hyperheuristic* untuk permasalahan Post Enrollment Course Timetabling (PE-CTT) diusulkan pada penelitian ini. *Self-Adaptive* digunakan pada penyeleksian LLH sementara *Tabu Search* dan *Simulated Annealing* digunakan pada move acceptance sehingga memenuhi kriteria Hiperheuristik yang memiliki penyeleksian LLH dan *Move acceptance*. Kesimpulan dari hibridisasi algoritma yaitu :

a. Hibridisasi ketiga algoritma berhasil diimplementasikan dengan memenuhi seluruh *hard constraints* dan dapat mengoptimasi PECTT pada dataset Socha dan ITC2007.
b. Hasil eksperimen pada dataset Socha menunjukkan bahwa algoritma *Self Adaptive Tabu Search Simulated Annealing* dapat mencapai solusi yang paling optimum pada instance yang memiliki kompleksitas yang paling sedikit. Dapat disimpulkan bahwa algoritma *Self Adaptive Tabu Search Simulated Annealing* memberikan hasil yang cukup baik.
c. Berbeda dengan hasil eksperimen pada dataset Socha, hasil eksperimen pada dataset ITC2007 menunjukkan bahwa 5 instance mencapai solusi yang lebih optimum daripada algoritma benchmark lain. Namun, algoritma *Self Adaptive Tabu Search Simulated Annealing* belum mampu mencapai solusi yang paling optimum dan tingkat *feasibility* kebanyakan kurang dari 100%. Perbedaan hasil eksperimen dikarenakan tingkat kompleksitas yang dimiliki oleh dataset ITC2007 lebih tinggi daripada tingkat kompleksitas dataset Socha.

6.2.Saran

Algoritma Self Adaptive-Tabu Search-dan Simulated Annealing telah berhasil diimplementasikan dalam kerangka Hiperheuristik namun pengimplementasian pada dataset ITC2007 belum memberikan hasil yang optimum dikarenakan tingkat feasibilitas yang masih kurang dan nilai solusi yang kurang optimum. Jika dibandingkan dengan algoritma benchmark yang telah dilakukan pada bab 5 uji coba dan analisis hasil, LLH yang digunakan lebih bervariasi dan pencarian lokalnya lebih kompleks. Ke depannya, diharapkan optimasi pada inisial solusi, modifikasi LLH, dan uji coba dengan parameter yang berbeda dilakukan sehingga dapat menambah *feasibility* dan menghasilkan solusi yang lebih optimal.

DAFTAR PUSTAKA

[1] M. Lindahl, A. J. Mason, T. Stidsen, and M. Sørensen, "Discrete Optimization A strategic view of University timetabling," *Eur. J. Oper. Res.*, vol. 266, no. 1, pp. 35–45, 2018.

[2] E. K. Burke and G. Kendall, *Search Methodologies*. 2014.

[3] R. Bai, E. K. Burke, G. Kendall, and B. Mccollum, "A Simulated Annealing Hyper-heuristic for University Course Timetabling," pp. 345–350, 2006.

[4] R. Ilyas and Z. Iqbal, "Study of Hybrid Approaches used for University Course Timetable Problem (UCTP)," in *The 10th IEEE Conference on Industrial Electronics and Applications (ICIEA 2015)*, 2015, pp. 696–701.

[5] L. Di Gaspero, B. Mccollum, and A. Schaerf, "The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)," no. Track 3, pp. 1–12, 2007.

[6] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *Eur. J. Oper. Res.*, vol. 140, no. 2, pp. 266–280, 2002.

[7] S. Leng, G. Kendall, and N. R. Sabar, "Improved local search approaches to solve the post enrolment course timetabling problem," *Eur. J. Oper. Res.*, vol. 261, no. 1, pp. 17–29, 2017.

[8] Q. K. Pan, M. Fatih Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Inf. Sci. (Ny).*, vol. 181, no. 12, pp. 2455–2468, 2011.

[9] W. Liu, T. Zhang, Y. Liu, N. Zhang, H. Tao, and G. Fu, "Improved artificial bee colony algorithm based on self-adaptive random optimization strategy," *Cluster Comput.*, vol. 4, pp. 1–10, 2018.

[10] 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.

[11] E. K. Burke, G. Kendall, and E. Soubeiga, “A Tabu-Search Hyperheuristic for Timetabling,” *J. Heuristics*, vol. 2003, no. 9, pp. 451–470, 2004.

[12] S. Naseem and J. Shengxiang, “A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling,” pp. 617–637, 2011.

[13] S. Ceschia, L. Di Gaspero, and A. Schaerf, “Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem,” *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1615–1624, 2012.

[14] E. K. Burke, B. Mccollum, and A. Meisels, “Discrete Optimization A graph-based hyper-heuristic for educational timetabling problems,” vol. 176, pp. 177–192, 2007.

[15] N. D. Thanh, “Solving Timetabling Problem Using Genetic and Heuristic Algorithms Nguyen Duc Thanh,” pp. 472–477, 2007.

[16] X. Li and M. Yin, “Self-adaptive constrained artificial bee colony for constrained numerical optimization,” *Neural Comput. Appl.*, vol. 24, no. 3–4, pp. 723–734, 2014.

[17] Ahsanul Marom, “Optimasi Penjadwalan Mata Kuliah Otomatis Menggunakan Algoritma Tabu-Simulated Annealing Hyper-Heuristics.” p. 189, 2018.

[18] 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.*, no. May 2014, pp. 413–427, 2005.

[19] L. W. Shen, H. Asmuni, and F. C. Weng, “A modified migrating bird optimization for university course timetabling problem,” *J. Teknol.*, vol. 72, no. 1, pp. 89–96, 2015.

[20] H. Asmuni, E. K. Burke, and J. M. Garibaldi, “Fuzzy multiple heuristic ordering for course timetabling,” *Proc. 2005 UK Work. Comput. Intell. UKCI 2005*, no. January 2005, pp. 302–309, 2005.

[21] S. Petrovic and R. Qu, “A Graph-Based Hyper Heuristic for Timetabling

Problems School of Computer Science and Information Technology University of Nottingham Computer Science Technical Report No . NOTTCS-TR-2004-9 A Graph-Based Hyper Heuristic for Timetabling Problems Edumnd K B," no. June, 2018.

[22] R. Lewis, "A time-dependent metaheuristic algorithm for post enrolment-based course timetabling," *7th Int. Conf. Pract. Theory Autom. Timetabling, PATAT 2008*, pp. 1–17, 2008.

[23] C. Nothegger, A. Mayer, A. Chwatal, and G. R. Raidl, "Solving the post enrolment course timetabling problem by ant colony optimization," *Ann. Oper. Res.*, vol. 194, no. 1, pp. 325–339, 2012.

Halaman ini sengaja dikosongkan

LAMPIRAN A. HASIL OPTIMASI PENJADWALAN DATASET SOCHA

Tabel A.1 Hasil Optimasi Penjadwalan Dataset Socha Small1

Day	Timeslot	Rooms				
		1	2	3	4	5
1	1				14	
	2	92	81			67
	3	52			20	
	4	29		85	11	33
	5	2	79			47
	6				61	
	7	74	3	30	26	93
	8	31			56	
	9					
2	10	58	100	97		49
	11	16	51	70	37	
	12			27	23	24
	13	91	35		7	
	14				64	
	15					
	16		42			
	17	32		82	95	36
	18					
3	19		5		71	
	20				38	44
	21		1		54	
	22	83				19
	23		65	89		
	24	45	10	9		48
	25	66				17
	26		43	59	80	
	27					
4	28	57				8
	29	78	75	34	98	
	30			13		
	31	68		50		77
	32		46			
	33	21			96	
	34	94	87		62	53
	35			60		6
	36					
5	37	22		25	90	72
	38					88
	39	69				
	40			41	4	40
	41		39			15
	42	73			12	55
	43	99			18	
	44	76	84	86	28	63
	45					

LAMPIRAN B. HASIL EKSPERIMEN SKENARIO A DATASET SOCHA

Tabel B.1 Hasil Eksperimen Skenario A Dataset Socha

	Small1	Small2	Small3	Small4	Small5	Medium1	Medium2	Medium3	Medium4	Medium5	Large
1	0	0	1	0	0	209	205	259	206	168	1006
2	1	1	0	3	0	207	216	258	222	235	920
3	0	1	3	2	0	225	218	244	202	173	1026
Min	0	0	0	0	0	207	205	244	202	168	920
Q1	0	0,5	0,5	1	0	208	210,5	251	204	170,5	963
Med	0	1	1	2	0	209	216	258	206	173	1006
Q3	0,5	1	2	2,5	0	217	217	258,5	214	204	1016
Max	2	1	3	3	0	225	218	259	222	235	1026

LAMPIRAN C. HASIL EKSPERIMEN SKENARIO B DATASET SOCHA

Tabel C.1 Hasil Eksperimen Skenario B Dataset Socha

	Small1	Small2	Small3	Small4	Small5	Medium1	Medium2	Medium3	Medium4	Medium5	Large
1	0	0	0	2	0	225	204	259	177	227	1021
2	1	0	1	1	4	203	238	254	220	158	1122
3	2	2	1	0	2	197	199	262	252	148	971
Min	0	0	0	0	0	197	199	254	177	148	971
Q1	0,5	0	0,5	0,5	1	200	201,5	256,5	198,5	153	996
Med	1	0	1	1	2	203	204	259	220	158	1021
Q3	1,5	1	1	1,5	3	214	221	260,5	236	192,5	1071,5
Max	2	2	1	2	4	225	238	262	252	227	1122

LAMPIRAN D. HASIL EKSPERIMEN SKENARIO C DATASET SOCHA

Tabel D.1 Hasil Eksperimen Skenario C Dataset Socha

	Small1	Small2	Small3	Small4	Small5	Medium1	Medium2	Medium3	Medium4	Medium5	Large
1	1	4	1	1	0	210	209	251	177	181	864
2	0	2	0	0	0	240	222	251	230	139	957
3	0	0	0	0	0	212	222	232	194	169	1142
Min	0	0	0	0	0	210	209	232	177	139	864
Q1	0	1	0	0	0	211	215,5	241,5	185,5	154	910,5
Med	0	2	0	0	0	212	222	251	194	169	957
Q3	0,5	3	0,5	0,5	0	226	222	251	212	175	1049,5
Max	1	4	1	1	0	240	222	251	230	181	1142

LAMPIRAN E. HASIL OPTIMASI PENJADWALAN DATASET ITC2007

Tabel E.1 Hasil Optimasi Penjadwalan Dataset ITC2007 Hidden17

Day	Timeslot	Room									
		1	2	3	4	5	6	7	8	9	10
1	1	35	91			46					
	2	87	6								
	3	25	43	53							
	4	7		52							
	5	72	100	3							
	6	36									
	7	62				96					
	8	93	99			23					
	9										
2	10	37	85								
	11	30									
	12	49				56	80				
	13	17	75	94		79					
	14	89	95								
	15	59	63								
	16	82	26			69					
	17	81	48			83					
	18		97								
3	19	74	90	11		65					
	20	14	24								
	21	67	45	39							
	22	8		92							
	23	86	1								
	24	61									
	25	34	42								
	26	47	13	54							
	27			12							
4	28	32	29			22	31				
	29	33				68					
	30	5	73								
	31	20	55								
	32	38	2								
	33	51	41			88					
	34	9	21								
	35	84	76			16					
	36	77									
5	37	27		19		28					
	38		57			18					
	39	58	66								
	40	4									
	41	78		40				64			
	42	44	50			10					
	43		15			71					
	44	98	60			70					
	45										

LAMPIRAN F. HASIL EKSPERIMEN SKENARIO A DATASET ITC2007

DTF = Distance to Feasibility. %Feas= Prosentase Feasibility .FS= Final Solution/Total Penalty

Tabel F.1 Hasil Eksperimen Skenario A Dataset ITC2007

Instance	1			2			3			Min DTF	Min %Feas	Min FS	Q1 FS	Med FS	Q3 FS	Max DTF	Max %Feas	Max FS
	DTF	%Feas	FS	DTF	%Feas	FS	DTF	%Feas	FS									
Early1	1603	84	1890	1769	81	1957	1581	84	1982	1581	81	1890	1923,5	1957	1969,5	1769	84	1982
Early2	2081	78	1820	1884	80	2044	1769	81	1806	1769	78	1806	1813	1820	1932	2081	80	2044
Early3	706	95	2538	444	97	2306	629	96	2646	444	95	2306	2422	2538	2592	706	97	2646
Early4	1402	89	2818	1402	89	2818	1401	89	2766	1401	87	2577	2671,5	2766	2792	1569	89	2818
Early5	787	87	1144	757	88	1166	899	85	1135	757	85	1135	1139,5	1144	1155	899	88	1166
Early6	721	89	1202	960	85	1176	880	86	1141	721	85	1141	1158,5	1176	1189	960	89	1202
Early7	815	88	1530	837	88	151	829	88	1418	815	88	1418	1466,5	1515	1522,5	837	88	1530
Early8	638	90	1667	574	91	1762	710	89	1615	574	89	1615	1641	1667	1714,5	710	91	1762
Late9	1508	84	2023	1535	85	1901	1517	84	1913	1508	84	1901	1907	1913	1968	1535	85	2023
Late10	1949	80	1981	1928	80	1733	2268	78	1733	1928	78	1733	1733	1733	1857	2268	80	1981
Late11	1477	90	2697	1226	91	2574	1026	93	2357	1026	90	2357	2465,5	2574	2635,5	1477	93	2697
Late12	1715	86	2870	1870	85	2743	2089	84	2696	1715	84	2696	2719,5	2743	2806,5	2089	86	2870
Late13	1024	84	1185	1066	84	1140	929	86	1155	929	84	1140	1147,5	1155	1170	1066	86	1185
Late14	877	87	1200	989	85	1041	966	85	1202	877	85	1041	1120,5	1200	1201	989	87	1202
Late15	634	90	1426	600	90	1434	563	91	1426	563	90	1426	1426	1426	1430	634	91	1434
Late16	173	97	1470	101	98	1526	279	95	1587	101	95	1470	1498	1526	1556,5	279	98	1587
Hidden17	0	100	1229	0	100	1207	0	100	1098	0	100	1098	1152,5	1207	1218,5	0	100	1229
Hidden18	1385	85	2000	1370	84	2000	1573	83	2004	1370	83	2000	2000	2000	2002	1573	85	2004
Hidden19	1978	85	2617	1893	86	2820	2124	83	2696	1893	83	2617	2656,5	2696	2758	2124	86	2820
Hidden20	614	95	2954	400	97	2990	454	96	2909	400	95	2909	2931,5	2954	2972	614	97	2990
Hidden21	589	91	1176	637	90	1242	473	92	1259	473	90	1176	1209	1242	1250,5	637	92	1259
Hidden22	2287	79	1628	2338	78	1734	2198	80	1688	2198	78	1628	1658	1688	1711	2338	80	1734
Hidden23	4844	77	3354	4279	80	3674	4601	79	3332	4279	77	3332	3343	3354	3514	4844	80	3674
Hidden24	1688	88	2909	1626	89	3047	1430	90	2950	1430	88	2909	2929,5	2950	2998,5	1688	90	3047

LAMPIRAN G. HASIL EKSPERIMEN SKENARIO B DATASET ITC2007

DTF = Distance to Feasibility. %Feas= Prosentase Feasibility .FS= Final Solution/Total Penalty

Tabel G.1 Hasil Eksperimen Skenario B Dataset ITC2007

Instance	1			2			3			Min DTF	Min %Feas	Min FS	Q1 FS	Med FS	Q3 FS	Max DTF	Max %Feas	Max FS
	DTF	%Feas	FS	DTF	%Feas	FS	DTF	%Feas	FS									
Early1	1480	85	2012	1572	84	1881	1616	83	1949	1480	83	1881	1915	1949	1980,5	1616	85	2012
Early2	1726	82	1866	1661	83	1879	1851	81	1760	1661	1851	1879	1813	1866	1872,5	1851	83	1879
Early3	646	95	3313	759	95	2730	615	96	2158	615	95	2158	2444	2730	3021,5	759	96	3313
Early4	1373	89	3089	1212	90	3013	1330	89	2744	1212	90	2744	2878,5	3013	3051	1373	90	3089
Early5	880	86	1165	826	86	1181	904	85	1134	826	85	1134	1149,5	1165	1173	904	86	1181
Early6	914	86	1172	960	85	1070	886	87	1085	886	85	1070	1077,5	1085	1128,5	960	87	1172
Early7	917	87	1452	736	89	1377	737	89	1312	736	87	1312	1344,5	1377	1414,5	917	89	1452
Early8	675	90	1608	644	90	1471	628	90	1557	628	90	1471	1514	1557	1582,5	675	90	1608
Late9	1484	85	1993	1392	86	2061	1626	84	2021	1392	84	2061	2007	2021	2041	1626	86	2061
Late10	2204	77	1763	1944	80	1766	2154	78	1738	1944	77	1738	1750,5	1763	1764,5	2204	80	1766
Late11	1199	91	2286	1359	91	2346	1305	91	2264	1199	91	2264	2275	2286	2316	1359	91	2346
Late12	1709	87	2684	1414	88	2837	1864	85	3096	1414	85	2684	2760,5	2837	2966,5	1864	88	3096
Late13	657	90	1207	812	87	1183	925	86	1158	657	86	1158	1170,5	1183	1195	925	90	1207
Late14	888	86	1138	1159	82	1136	1067	83	1105	888	82	1105	1120,5	1136	1137	1159	86	1138
Late15	731	88	1511	461	93	1392	556	91	1510	461	88	1392	1451	1510	1510,5	731	93	1511
Late16	282	95	1456	94	98	1516	188	97	1485	94	95	1456	1470,5	1485	1500,5	282	98	1516
Hidden17	0	100	1216	0	100	1135	0	100	1488	0	100	1135	1175,5	1216	1352	0	100	1488
Hidden18	1339	85	2027	1337	85	1976	1536	83	1623	1337	83	1623	1799,5	1976	2001,5	1536	85	2027
Hidden19	1793	86	2920	1967	85	2858	2315	83	2601	1793	83	2601	2729,5	2858	2889	2315	86	2920
Hidden20	396	96	2940	399	97	3078	375	97	3207	375	96	2940	3009	3078	3142,5	399	97	3207
Hidden21	685	89	1163	622	90	1182	643	90	1203	622	89	1163	1172,5	1182	1192,5	685	90	1203
Hidden22	2236	79	1846	1994	81	1806	2109	80	1784	1994	79	1784	1795	1806	1826	2236	81	1846
Hidden23	4723	78	3403	4471	79	3275	4970	77	3611	4471	77	3275	3339	3403	3507	4970	79	3611
Hidden24	1583	88	2988	1852	87	3017	1990	86	2864	1583	86	2864	2926	2988	3002,5	1990	88	3017

LAMPIRAN H. HASIL EKSPERIMEN SKENARIO C DATASET ITC2007

DTF = Distance to Feasibility. %Feas= Prosentase Feasibility. FS= Final Solution/Total Penalty

Tabel I.1 Hasil Eksperimen Skenario C Dataset ITC2007

Instance	1			2			3			Min DTF	Min %Feas	Min FS	Q1 FS	Med FS	Q3 FS	Max DTF	Max %Feas	Max FS
	DTF	%Feas	FS	DTF	%Feas	FS	DTF	%Feas	FS									
Early1	1538	84	1868	1549	84	1960	1667	83	1963	1538	83	1868	1914	1960	1961,65	1667	84	1963
Early2	1903	80	1789	1974	79	1816	2006	79	1805	1903	79	1789	1797	1805	1810,5	2006	80	1816
Early3	1029	93	2926	817	94	2848	862	94	2735	817	93	2735	2791,5	2848	2887	1029	94	2926
Early4	1468	88	3043	1636	87	3364	1034	91	3145	1034	87	3043	3094	3145	3254,5	1636	91	3364
Early5	883	86	1208	915	86	1133	921	85	1142	883	85	1133	1137,5	1142	1175	921	86	1208
Early6	837	88	1112	978	85	1086	849	87	1169	837	85	1086	1099	1112	1140,5	978	88	1169
Early7	943	86	1452	896	87	1413	712	90	1441	712	86	1413	1427	1441	1446,5	943	90	1452
Early8	640	90	1531	578	91	1553	747	89	1610	578	89	1531	1542	1553	1581,5	747	91	1610
Late9	1254	87	2098	1278	87	2012	1417	85	1980	1254	85	1980	1996	2012	2055	1996	2012	2055
Late10	1980	80	1717	2003	80	1739	2255	78	1789	1980	78	1717	1728	1739	1764	2255	80	1789
Late11	1201	91	2866	1374	90	3038	1576	89	3080	1201	89	2866	2952	3038	3059	1576	91	3080
Late12	1818	86	2914	2192	83	3237	1720	86	3109	1720	83	2914	3011,5	3109	3173	2192	86	3173
Late13	956	86	1250	1082	84	1125	953	86	1156	953	84	1125	1140,5	1156	1203	1082	86	1250
Late14	971	85	1170	881	87	1142	1125	83	999	881	83	999	1070,5	1142	1156	1125	87	1170
Late15	740	89	1542	665	89	1432	721	89	1505	665	89	1432	1468,5	1505	1523,5	740	89	1542
Late16	300	95	1653	415	93	1767	262	95	1613	262	93	1613	1633	1653	1710	415	95	1767
Hidden17	0	100	1003	0	100	1211	0	100	1305	0	100	1003	1107	1211	1258	0	100	1305
Hidden18	1487	84	1985	1515	83	1890	1786	80	1821	1487	80	1821	1855,5	1890	1937,5	1786	84	1985
Hidden19	1847	86	3003	2134	85	2939	1910	86	2954	1847	85	2939	2946,5	2954	2978,5	2134	85	3003
Hidden20	346	97	2918	507	96	3029	400	97	3046	346	96	2918	2973,5	3029	3037,5	507	97	3046
Hidden21	708	89	1165	496	92	1199	582	91	1232	496	89	1165	1182	1199	1215,5	708	92	1232
Hidden22	2303	79	1661	2088	80	1808	2554	76	1619	2088	76	1619	1640	1661	1734,5	2554	80	1808
Hidden23	4271	80	3819	4635	78	3586	4492	79	3505	4271	80	3505	3545,5	3586	3702,5	4635	80	3819
Hidden24	1985	86	2830	1673	88	2871	1912	86	2991	1673	86	2830	2850,5	2871	2931	1985	88	2991

LAMPIRAN I. HASIL EKSPERIMEN SKENARIO D DATASET ITC2007

DTF = Distance to Feasibility. %Feas= Prosentase Feasibility .FS= Final Solution/Total Penalty

Tabel J.1 Hasil Eksperimen Skenario D Dataset ITC2007

Instance	1			2			3			Min DTF	Min %Feas	Min FS	Q1 FS	Med FS	Q3 FS	Max DTF	Max %Feas	Max FS
	DTF	%Feas	FS	DTF	%Feas	FS	DTF	%Feas	FS									
Early1	1712	83	1703	1550	84	1765	1587	84	1841	1550	83	1703	1734	1765	1803	1712	83	1841
Early2	1806	81	1749	1852	80	1826	1948	80	1756	1806	80	1749	1752,5	1756	1791	1948	81	1826
Early3	1081	93	1835	770	95	1717	442	97	1954	442	93	1717	1776	1835	1894,5	1081	97	1954
Early4	1123	91	2076	1713	86	2127	1331	90	1897	1123	86	1897	1986,5	2076	2102,5	1713	91	2127
Early5	784	87	1143	884	86	1083	812	87	1090	784	86	1083	1086,5	1090	1116,5	884	87	1143
Early6	926	86	1133	899	86	1078	820	87	1125	820	86	1078	1101,5	1125	1129	926	87	1133
Early7	912	87	1226	1085	85	1156	825	87	1329	825	85	1156	1191	1226	1277,5	1085	87	1329
Early8	658	90	1272	510	92	1125	844	87	1125	510	87	1125	1125	1125	1198,5	844	92	1272
Late9	1442	85	1980	1597	84	1916	1443	86	1780	1442	84	1780	1848	1916	1948	1597	86	1980
Late10	2092	78	1710	1955	80	1756	1869	80	1882	1869	78	1710	1733	1756	1819	2092	80	1882
Late11	1564	89	2086	1414	89	2121	1106	92	2146	1106	89	2086	2103,5	2121	2133,5	1564	92	2146
Late12	1782	86	2486	1843	86	2359	1553	88	2055	1553	86	2055	2207	2359	2422,5	1843	88	2486
Late13	984	85	1071	877	87	1191	1085	84	1100	877	84	1071	1085,5	1100	1145,5	1085	87	1191
Late14	994	85	1027	954	85	1118	1053	84	1063	954	84	1027	1045	1063	1090,5	1053	85	1118
Late15	444	93	1264	611	91	1331	484	92	1282	444	91	1264	1273	1282	1306,5	611	93	1331
Late16	296	95	1100	269	95	983	228	96	890	228	95	890	936,5	983	1041,5	296	96	1100
Hidden17	0	100	412	0	100	536	0	100	563	0	100	412	474	536	549,5	0	100	563
Hidden18	995	88	1890	1607	81	1710	1737	81	1785	995	81	1710	1747,5	1785	1837,5	1737	81	1890
Hidden19	2049	84	2412	2121	84	2359	2371	83	2449	2049	83	2359	2385,5	2412	2430,5	2371	84	2449
Hidden20	408	97	2428	484	96	2459	354	97	2395	354	96	2395	2411,5	2428	2443,5	484	97	2443,5
Hidden21	731	89	1116	577	91	1136	589	91	1117	577	89	1116	1116,5	1117	1126,5	731	91	1136
Hidden22	2259	79	1710	2271	79	1696	2068	81	1762	2068	79	1696	1703	1710	1736	2271	81	1762
Hidden23	4421	80	3287	4890	77	3080	4524	78	3033	4421	77	3033	3056,5	3080	3183,5	4890	80	3287
Hidden24	1917	86	2293	1432	90	2673	1730	88	2229	1432	86	2229	2261	2293	2483	1917	90	2673

BIOGRAFI PENULIS

Kartika Maulida Hindrayani, lahir di Kota Surabaya pada 9 September 1992. Penulis merupakan anak pertama dari dua bersaudara. Penulis menempuh pendidikan formal di SD Muhammadiyah 4 Surabaya pada tahun 1998 - 2004, SMPN 19 Surabaya pada tahun 2004 - 2007, SMAN 16 Surabaya pada tahun 2007 - 2010. Penulis melanjutkan studi S1 di Jurusan Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2010. Penulis aktif sebagai staff Pengembangan Sumber Daya Mahasiswa (PSDM) Himpunan Mahasiswa Sistem Informasi (HMSI) ITS dan staff PSDM Badan Eksekutif Mahasiswa (BEM) ITS pada tahun 2011-2013. Pada tahun 2015 penulis menyelesaikan studi S1. Selama setahun penulis bekerja di sebuah perusahaan *software house* sebagai *application support*. Pada tahun 2017 penulis melanjutkan ke jenjang S2 di program studi yang sama yaitu Departemen Sistem Informasi ITS. Laboratorium yang menangani penelitian ini adalah Laboratorium Rekayasa Data dan Intelegensia Bisnis (RDIB). Penulis menyelesaikan studi S2 pada tahun 2020. Kritik dan saran yang membangun dapat dikirim ke kartikamaulida@gmail.com.