TESIS - IS185401

# PENJADWALAN UJIAN MENGGUNAKAN METODE HYPER-HEURISTIK DENGAN ALGORITMA PREY PREDATOR DAN SIMULATED ANNEALING

RUSDI HAMIDAN
NRP. 05211650010013

DOSEN PEMBIMBING
Ahmad Muklason, S.Kom , M.Sc., Ph.D
NIP: 198203022009121000

PROGRAM MAGISTER
DEPARTEMEN SISTEM INFORMASI
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

## SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertandatangan di bawah ini:

| | |
|---|---|
| Nama | : Rusdi Hamidan |
| Tempat/ Tanggal Lahir | : Semarang, 21 November 1991 |
| NRP | : 05211650010013 |
| Program Studi | : S2 Sistem Informasi |
| Judul Tesis | : Penjadwalan Ujian Menggunakan Metode Hyper-Heuristik Dengan Algoritma Prey-Predator dan Simulated Annealing |

Saya yang bertanda tangan di bawah ini dengan sebenarnya menyatakan bahwa tesis ini saya susun tanpa tindakan plagiarisme sesuai dengan peraturan yang berlaku di Institut Teknologi Sepuluh November Surabaya.

Jika di kemudian hari ternyata saya melakukan tindakan plagiarisme, saya akan bertanggung jawab sepenuhnya dan menerima sanksi yang dijatuhkan oleh Institut Teknologi Sepuluh November Surabaya.

Surabaya, 16 Juli 2020


Rusdi Hamidan

# LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
**Magister Sistem Informasi (M.Kom.)**
di
**Institut Teknologi Sepuluh Nopember**

Oleh:
**Rusdi Hamidan**
**NRP: 05211650010013**

Tanggal Ujian: 03 Agustus 2020
Periode Wisuda: September 2020

Disetujui oleh:
**Pembimbing:**

Ahmad Muklason, S.Kom., M.Sc., Ph.D.
NIP: 198203022009121009 .....

**Penguji:**

Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D.
NIP: 198201202005012001 .....

Faizal Mahananto, S. Kom, M.Eng., Ph.D.
NIP: 198510312019031009 .....

Surabaya, 24 Agustus 2020
Kepala Departemen Sistem Informasi
Fakultas Teknologi Elektro dan Informatika Cerdas

Dr. Mujahidin, S.T., M.T.
NIP. 197010102003121001

# Penjadwalan Ujian Menggunakan Metode Hyper-Heuristik Dengan Algoritma Prey-Predator dan Simulated Annealing


Nama mahasiswa : Rusdi Hamidan
NRP : 05211650010013
Pembimbing : Ahmad Muklason, S.Kom , M.Sc., Ph.D


## ABSTRAK


Penjadwalan Ujian atau *Exam Timetabling Problem* (ETP) adalah sebuah permasalahan yang terjadi pada pihak universitas. Penyelesaian terhadap permasalahan ETP tersebut melibatkan metode pencarian komputasional untuk memperoleh hasil. Dalam proses tersebut, apabila menggunakan algoritma eksak akan membutuhkan waktu yang banyak untuk mencapai solusi yang optimal. Penjadwalan ujian pada dasarnya merupakan pengalokasian jadwal ke dalam sebuah ruangan di waktu tertentu. Beberapa peneliti terdahulu mengembangkan metode hyper-heuristik untuk memperoleh solusi yang diharapkan mampu memberikan hasil yang mendekati optimal.

Dalam penelitian ini akan menggunakan dataset ITC 2007 untuk menemukan solusi generic yang mendekati optimal. Algoritma Prey-Predator dipilih sebagai strategi untuk memilih Low Level Heuristic (LLH) dan Simulated Annealing dipilih (SA) sebagai strategi move-acceptance terhadap ETP.

Hasil dari penelitian ini menunjukkan bahwa dengan menggabungkan algoritma Prey-Predator dengan Simulated Annealing, performanya lebih unggul dibanding dengan algoritma Simple Random Hill Climbing, Simple Random Simulated Annealing, dan Prey Predator pada 4 instance dari total 8 instance.



Kata Kunci : *Prey Predator, Simulated Annealing, Examination Timetabling Problem, Hyflex, Hyper-Heuristic, ITC 2007*

# Examination Timetabling Problem Using Hyperheuristic Method with Prey Predator – Simulated Annealing Algorithm


| | |
|---|---|
| By | : Rusdi Hamidan |
| Student Identity Number | : 05211650010013 |
| Supervisor | : Ahmad Muklason, S.Kom , M.Sc., Ph.D |


## ABSTRACT


Exam Scheduling or Exam Timetabling Problem (ETP) is a problem that occurs at the university. The solution to the ETP problem involves computational search methods to obtain results. In the process, when using exact algorithm it will require a lot of time to achieve an optimal solution. Scheduling an exam is basically allocating a schedule into a room at particular time. Several previous researchers developed a hyper-heuristic method to obtain solutions that are expected to provide results that are close to optimal.

In this research, ITC 2007 dataset will be used to find generic solutions that are close to optimal. Prey-Predator Algorithm (PPA) was chosen as a strategy to choose Low Level Heuristic (LLH) and Simulated Annealing (SA) as move-acceptence strategy for ETP.

The result of this study indicate by combining the Prey-Predator algorithm with Simulated Annealing, its performance is superior than Simple Random Hill Climbing, Simple Random Simulated Annealing, and Prey-Predator on 4 out of total 8 instances.



Keyword : *Prey Predator, Simulated Annealing, Examination Timetabling Problem, Hyflex, Hyper-Heuristic, ITC 2007*

# KATA PENGANTAR

Puji syukur kepada Allah SWT, karena dengan rahmat dan ridho-Nya penulis dapat menyelesaikan laporan Tesis yang berjudul "Penjadwalan Ujian Menggunakan Metode Hiperheuristik dengan Algoritma Prey Predator dan Simulated Annealing" sebagai salah satu syarat kelulusan Program Magister Departemen Sistem Informasi. Pada proses pengerjaan sampai terselesaikannya Tesis ini, penulis telah mendapatkan bantuan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Dr. Mudjahidin, S.T., M.T. selaku Kepala Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
2. Bapak Tony Dwi Susanto, S.T., M.T., Ph.D. selaku Ketua Prodi S2 Departemen Sistem Informasi Institut Teknologi Sepuluh Nopember.
3. Bapak Ahmad Muklason S.Kom., M.Sc., Ph.D. selaku Pembimbing Tesis yang telah meluangkan waktu dan dengan sabar membimbing penulis sampai menyelesaikan Tesis ini.
4. Nur Aini Rakhmawati, S.Kom., M.Sc.Eng., Ph.D. dan Retno Aulia Vinarti, S.Kom., M.Kom., Ph.D. selaku dosen penguji sidang Tesis.
5. Seluruh staf akademik dan administrasi di Departemen Sistem Informasi ITS atas informasi dan bantuan yang telah diberikan.
6. Keluarga tercinta, terutama Bapak Harry Soetanto dan Ibu Mutmainah yang perannya tak terkira dalam doa dan dukungan baik secara moral maupun finansial.
7. Teman seperjuangan Deny Hermansyah
8. Rekan satu pembimbing, Tim Laboratorium RDIB Sasmi Hidayatul Yulianing Tyas, Nisa Dwi Angresti atas segala bantuan, informasi, dan waktu yang diberikan kepada penulis serta semangat untuk berjuang bersama sehingga tesis ini dapat diselesaikan.

9. Keluarga dari Magister Sistem Informasi Institut Teknologi Sepuluh Nopember lainnya atas segala dukungan dan motivasi serta kenangan yang akan diingat oleh penulis sebagai bagian dari keluarga ini.
10. Serta semua pihak lainnya yang telah membantu penulis baik secara langsung maupun tidak langsung hingga terselesaikannya laporan Tesis ini.

Penulis memohon maaf apabila masih terdapat banyak kekurangan pada laporan Tesis ini dan apabila terdapat kritik serta saran yang membangun dari pembaca dapat dikirimkan melalui email rusdi.hamidan@gmail.com. Semoga laporan ini dapat bermanfaat.

Surabaya, 6 Juli 2020

Penulis

# DAFTAR ISI

# DAFTAR TABEL

# DAFTAR GAMBAR

# DAFTAR KODE

*Halaman ini sengaja dikosongkan*

# BAB 1
# PENDAHULUAN

Bab ini membahas mengenai latar belakang permasalahan, perumusan masalah, tujuan dan manfaat penelitian, kontribusi penelitian, batasan dan sistematika penelitian.

## 1.1 Latar Belakang Masalah

Sudah sejak lama manusia berusaha menyelesaikan permasalahan dengan hasil terbaik. Secara contoh adalah pembelian stok pada toko agar memenuhi kuota pembelian. Disini kita harus memutuskan jumlah barang yang disimpan dan kapan harus dibeli. Selain itu, terdapat banyak permasalahan optimasi dimana kita mengoptimasi uang, waktu, barang, dan sebagainya dalam beberapa batasan. Teknik optimasi sendiri berada dalam model riset operasi.

Riset operasional merupakan suatu cabang ilmu pengetahuan yang berkembang saat Perang Dunia ke II (Bustani, 2005). Riset operasi dapat didefinisikan sebagai penggunaan metode kuantitatif untuk membantu analisis dan mengambil keputusan dalam merancang, menganalisis, dan meningkatkan kinerja dari suatu sistem (Carter, 2018). Dengan menggunakan riset operasi sebagai salah satu alat untuk menentukan keputusan-keputusan penting, penggunaan sumber daya menjadi minimal sedangkan keuntungan yang didapat maksimal.

Riset operasional telah banyak digunakan dalam berbagai bidang seperti transportasi, olahraga, pemerintahan, manufaktur, dan pendidikan. Contohnya pada sektor pertanian, dengan populasi meledak yang mengakibatkan kekurangan pangan, setiap negara menghadapi masalah alokasi lahan yang optimal untuk berbagai tanaman yang sesuai dengan kondisi iklim dan fasilitas yang tersedia. Contoh pada rumah sakit, penjadwalan untuk pengembangan pemanfaatan ruang operasi dan meningkatkan kepuasan pasien (Wang, 2008). Contoh pada pembangunan, pembuatan model jam istirahat untuk pekerja konstruksi, dimana tujuannya adalah meminimalisir waktu yang dibutuhkan untuk menyelesaikan pekerjaan dan meminimalisir tenaga tambahan yang digunakan pekerja karena

bekerja lebih lama dari durasi kerja maksimum (Hsie, 2009). Salah satu topik yang sering dibahas pada riset operasional adalah penjadwalan.

Penjadwalan merupakan kegiatan pengalokasian sumber-sumber atau mesin-mesin yang ada untuk menjalankan sekumpulan tugas dalam jangka waktu tertentu (Baker, 1974). Jadwal biasanya memberitahu kita kapan suatu hal atau acara akan terjadi, jadwal menunjukkan kita rencana dari suatu aktivitas yang akan dilaksanakan pada waktu tertentu. Jadwal sangat berguna contohnya membantu penumpang mengetahui kapan bis akan berangkat, membantu mereka merencanakan jadwal mereka sendiri (Baker, 2009).

Terdapat berbagai macam implementasi pada penjadwalan, seperti penjadwalan sekolah, penjadwalan universitas, penjadwalan tempat kerja, penjadwalan penerbangan, penjadwalan transportasi umum, dan penjadwalan acara olahraga (Doulaty, 2013). Diantara penjadwalan tersebut terdapat penjadwalan dalam dunia pendidikan seperti penjadwalan pelajaran dan penjadwalan ujian. Penjadwalan pada dunia pendidikan merupakan tugas yang sulit karena memiliki banyak kendala yang harus dipenuhi untuk mendapatkan solusi yang layak.

Permasalahan pada penjadwalan pendidikan dikategorikan menjadi tiga, penjadwalan sekolah, penjadwalan pelajaran, dan penjadwalan ujian (Schaerf, 1999). Penelitian ini akan berfokus pada penjadwalan ujian. Penjadwalan ujian dapat didefinisikan sebagai permasalahan pencarian tugas yang terbaik. Tujuan umum dari penjadwalan ujian adalah untuk memenuhi semua *hard constraint* dan meminimalisir penalty yang diberikan *soft constraint* yang dapat bervariasi antar universitas.

Pada penjadwalan sendiri memiliki aturan (*constraint*) yang tidak boleh dilanggar, biasanya dikategorikan menjadi dua, yaitu *hard constraint* dan *soft constraint*. *Hard constraint* sendiri tidak boleh dilanggar dalam situasi apapun yang apabila dilanggar akan membuat sebuah solusi tidak layak, contohnya ujian yang saling bertabrakan atau pelajar tidak bisa melaksanakan ujian lebih dari satu pada waktu yang sama. Soft Constraint merupakan batasan yang diinginkan tetapi tidak menyebabkan masalah apabila dilanggar. Pelanggaran pada *soft constraint* biasanya akan menyebabkan pengurangan nilai atau pinalti pada penjadwalan tersebut. Contoh *soft constraint* pada penjadwalan ujian adalah memberikan waktu yang cukup antara ujian agar pelajar dapat melakukan waktu untuk melakukan pembelajaran (Burke, 2009).

Penjadwalan ujian dikategorikan sebagai permasalahan NP-complete (Even, 1975). NP merupakan singkatan dari *Nondeterministic Polynomial time* merupakan permasalahan yang membutukan waktu komputasi yang proporsional (Carter, 2018). Di dalam permasalahan NP terdapat permasalahan yang dinamakan NP-complete, permasalahan yang paling sulit di dalam permasalahan NP. Ini artinya bahwa jumlah komputasi yang sangat besar dibutuhkan untuk menyelesaikan permasalahan dengan berbagai ukuran, oleh karenanya masalah tersebut disebut dengan masalah yang tidak bisa diselesaikan atau *intractable*.

Agar dapat memberikan rumusan masalah dan dataset pada permasalahan penjadwalan ujian termasuk masalah yang sebenarnya pada permasalahan penjadwalan ujian, beberapa penelitian telah berbagi datasetnya. Dua benchmark dataset yang telah banyak digunakan para peneliti adalah Carter (Carter, 1996) dan International Timetabling Competition 2007 atau ITC 2007 (Gasperro, 2007). Penelitian ini akan berfokus pada dataset ITC 2007 dikarenakan datanya yang lebih kompleks dan lebih mendekati dengan permasalahan di dunia nyata.

Terdapat banyak metode untuk menyelesaikan permasalahan pada penjadwalan ujian, heuristik (Laporte, 1984), simulated annealing (Eglese, 1987), algoritma genetika (Corne, 1993), dan metode lainnya. Akibat kompleksitas dataset, maka penyelesaiannya akan lebih tepat apabila menggunakan metode heuristik daripada menggunakan metode eksak.

Pada penelitian terkait survey pemecahan penjadwalan ujian (Que et al, 2009), terdapat dua heuristik yang sering digunakan untuk pemecahan penjadwalan ujian, yaitu *meta-heuristic* dan *hyper-heuristic*. Meta-heuristic adalah metode yang paling banyak digunakan pada permasalahan penjadwalan ujian. Metode hyper-heuristic bertujuan untuk mengatasi keterbatasan metode meta-heuristic.

Gagasan dasar hyper-heuristic adalah untuk mengembangkan algoritma yang lebih umum dibandingkan dengan meta-heuristic yang pada umumnya memerlukan parameter tuning untuk memecahkan setiap permasalahan. Dalam meta-heuristic, pengetahuan problem domain diperlukan untuk parameter tuning (Cowling, 2001). Untuk mengatasi masalah ini hyper-heuristic muncul dengan bekerja pada level yang lebih tinggi, contoh *low-level heuristic search space* daripada langsung ke *solution search space*.

Pada peneletian review tentang pendekatan hyper-heuristic pada permasalahan penjadwalan ujian (Peter, 2011), menyebutkan runner-up dari kompetisi ITC 2007 kedua (Gogos, 2012) menjelaskan tahap pengembangan yang dilakukan setelah melakukan inisialisasi solusi, dimana simulated annealing diterapkan agar tidak terjebak dari *local optimum*. Pada penelitiannya, inisialisasi solusi membutuhkan waktu sekitar dua puluh persen dan sisanya digunakan untuk tahap pengembangan.

Algoritma *Prey-Predator* diterapkan pada examination timetabling menggunakan *meta-heursitic* (Tilahun, 2019). Dimana konsep dari algoritmanya terinspirasi dari hubungan *prey* dan predator. Insial solusi yang sudah diciptakan dipisah menjadi tiga kategori, solusi yang baik *best prey*, solusi yang buruk *predator*, dan sisanya *ordinary prey*. *Best prey* akan terfokus pada eskploitasi di sekitarnya, sedangkan *predator* akan mengeksplorasi area di sekitarnya untuk mencari solusi yang menjanjikan. *Ordinary prey* akan terpengaruh terhadap dua perilaku eksplorasi dan eksploitasi ini. Algoritma ini diujikan di dataset carter dengan menghasilkan nilai cukup baik. Dikarenakan algoritma *best prey* yang terfokus pada eksploitasi dan tidak tergantung dengan sekitarnya, maka *best prey* dapat memiliki kemungkinan untuk terjebak dalam *local optimum*.

Performa prey-predator terpengaruh terhadap jumlah best prey, predator, dan ordinary prey. Jumlah best prey yang terlalu banyak maka akan berpengaruh pada algoritma yang berfokus pada eksploitasi solusi, sedangkan predator yang terlalu banyak menyebabkan algoritma berfokus pada eksplorasi. Permasalahan ini kemudian dipecahkan oleh Tilahun, dimana penelitian tersebut mencari rasio antara ketiganya menggunakan hyper-heuristic (Tilahun, 2017).

Pada penelitian ini akan menyelesaikan permasalahan examination penjadwalan ujian dengan menggabungkan hyper-heuristic prey-predator dengan simulated annealing. Dataset yang digunakan adalah dataset ITC 2007. Penelitian ini bertujuan untuk mengevaluasi metode yang diusulkan dengan pengujian terhadap dataset ITC 2007.

## 1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, diketahui bahwa terdapat permasalahan dalam menyelesaikan ETP 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 permasalahan penjadwalan ujian masih memerlukan pengembangan lebih lanjut, sehingga rumusan masalah yang ingin dijawab yaitu :

1. Bagaimana pengaplikasian metode hyper-heuristic yang berbasis algoritma Prey-Predator sebagai strategi pemilihan low-level heuristic dan Simulated Annealing sebagai *move acceptance* pada permasalahan ETP?
2. Bagaimana performa dari metode hyper-heuristic yang berbasis algoritma Prey-Predator dengan Simulated Annealing yang diusulkan pada permasalahan ETP?

## 1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dari penelitian ini adalah menyelesaikan permasalahan *Examination Timetabling Problem* dengan benchmark dataset ITC 2007 agar dihasilkan solusi penjadwalan dengan nilai pelanggaran yang minimal. Permasalahan tersebut akan diselesaikan dengan penerapan algoritma Prey-Predator dan Simulated Annealing pada aplikasi yang telah dibuat dalam ruang lingkup Hyper-Heuristic.

## 1.4 Manfaat Penelitian

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

## 1.5 Kontribusi Penelitian

Penelitian ini diharapkan menghasilkan kontribusi secara teoritis dan praktis. Kontribusi dibahas sebagai berikut:

1. Kontribusi teoritis dalam penelitian ini yaitu mengkombinasikan algoritma Prey-Predator dan Simulated Annealing untuk menghasilkan solusi penjadwalan ujian yang lebih baik. Evaluasi kinerja dari metode yang diusulkan dengan pendekatan Hyper-Heuristics juga akan disajikan. Kontribusi penelitian ini didapatkan dengan penggabungan metode Simulated Annealing agar algoritma best prey dari Prey-Predator tidak terjebak dalam *local optimum*.
2. Sedangkan untuk kontribusi praktis dari penelitian ini adalah mengembangkan aplikasi yang telah dibuat sebelumnya pada domain ETP. Hal ini dapat membuat universitas melakukan otomasi proses penjadwalan sehingga proses penjadwalan ujian menjadi lebih efektif.

## 1.6 Batasan Penelitian

Penelitian ini memiliki beberapa batasan masalah sehingga penelitian ini tetap fokus pada algoritma yang dapat digunakan untuk meningkatkan hasil pada pengujian permasalahan penjadwalan ujian. Beberapa batasan masalah tersebut yaitu :

1. Dataset yang digunakan merupakan dataset ITC 2007
2. Permasalahan yang akan diselesaikan adalah permasalahan penjadwalan ujian (*Examination Timetabling Problem*)

## 1.7 Sistematika Penulisan

Sistematika penulisan laporan proposal penelitian ini adalah sebagai berikut :

**Bab 1: Pendahuluan**

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

**Bab 2: Kajian Pustaka**

Bab ini berisi kajian terhadap teori dan penelitian-penelitian terdahulu yang terkait. Kajian pustakan ini untuk memperkuat dasar-dasar dilakukannya penelitian.

**Bab 3: Metodologi Penelitian**

Bab ini berisi mengenai rancangan penelitian, lokasi dan tempat penelitian, serta tahapan-tahapan sistematis yang digunakan dalam penelitian.

**Bab 4: Riset Pendahuluan**

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

**Bab 5: Desain dan Implementasi**

Bab ini berisikan mengenai penjelasan desain tentang model proses pengembangan metode. Setelah desain selesai, dilakukan implementasi ke dalam bahasa pemgrograman.

**Bab 6: Hasil Implementasi dan Analisis**

Bab ini berisikan mengenai penjelasan analisis hasil dari uji coba yang dilakukan dalam penelitian ini. Analisis dilakukan dengan menguji hasil dari uji coba untuk mencari solusi optimal untuk kombinasi antara *Prey Predator* sebagai metode dalam mekanisme seleksi Low Level Heuristic (LLH) dan metode metode *Simulated Annealing* sebagai metode dalam mekanisme move acceptance.

**Bab 7: Kesimpulan dan Saran**

Bab ini berisikan mengenai penjelasan kesimpulan dari tujuan yang ingin dicapai serta saran untuk perbaikan terhadap kelemahan yang ditemukan dari hasil penelitian. Kesimpulan dan saran diambil berdasarkan seluruh proses penelitian yang telah dilakukan dalam penelitian ini.

**Daftar Pustaka**

Berisi daftar referensi yang digunakan dalam penelitian ini, baik jurnal ataupun buku.

*Halaman ini sengaja dikosongkan*

# BAB 2
# KAJIAN PUSTAKA

Bab ini membahas mengenai kajian pustaka dan dasar teori yang mendukung dalam pengerjaan penelitian. Kajian pustaka ini selanjutnya akan dibangun sebagai landasan dalam melakukan penelitian ini.

## 2.1 Penelitian Terdahulu (*State-of-the-art*)

Analisis penelitian sebelumnya menjelaskan state-of-art dari pendekatan prey-predator untuk penyelesaian beberapa problem domain dan juga state-of-art dari penelitian tentang problem domain ETP. Terdapat empat penelitian sebelumnya yang membahas tentang implementasi metode prey-predator ke dalam beberapa problem domain dan dua penelitian sebelumnya yang membahas tentang metode hyper-heuristik pada ETP. Analisis penelitian sebelumnya mengeskplorasi tentang pengembangan hyper-heuristic untuk menentukan parameter yang lebih baik, penggunaan seleksi dan move acceptance yang baru pada penelitian.

Pada penelitian ini (Asaju, 2015), dibahas mengenai hyper-heuristik pada ETP dengan menggunakan metode Artificial Bee Colony (ABC) yang telah dimodifikasi. Dalam penelitian ini metode tersebut dinamakan Hybrid Artificial Bee Colony (HABC) untuk permasalahan uncapacitated examination timetabling dengan menggunakan dataset Carter. Dalam metode HABC tersebut dikembangkan simple local search technique (SLST) pada operator lebah metode ABC dan mengganti konsep lebah pengintai dengan strategi Harmony Search Algorithm (HSA). Metode SLST sendiri digunakan untuk mencari kearah local optimum dengan perbedaan penambahan metode move, swap, dan kempechain setelah solusi dibuat. Sedangkan metode HSA sendiri digunakan untuk exploration agar tidak terjebak dalam local optimum. Dalam implementasinya, performa metode HABC melebih metode sebelumnya ABC dan juga beberapa metode lainnya, seperti Ant Colony dan Hybrid Particle Swarm Optimization.

Adapun penelitian yang hanya terfokus pada pengembangan move-acceptance. Penelitian ini (Bykov and Petrovic, 2016) mengembangkan metode Hill

Climbing (HC) menjadi Step Counting Hill Climbing (SCHC). Konsep dari SCHC adalah menggunakan iterasi ke dalam HC agar bisa mendapatkan hasil yang lebih baik. Parameter yang digunakan dalam penelitian ini disebut cost bound (Bc). Bc mendefinisikan batasan nilai yang bisa diterima. Contohnya, pada setiap iterasi algoritma menerima solusi yang memiliki nilai Bc lebih rendah (solusi lebih baik) dari Bc yang sudah ditetapkan dan menolak nilai Bc yang lebih tinggi (solusi lebih buruk) dari Bc yang sudah ditetapkan. SCHC dimulai menggunakan initial solution yang acak dan nilai Bc sama dengan nilai awal cost function. Kemudian algoritma mulai berjalan dengan iterasi (nc), jika nc mencapai batasan iterasi (Lc), maka Bc akan diperbarui. Penelitian ini kemudian diterapkan pada ETP dan dibandingkan performanya dengan metode SA dan Great Deluge (GD) pada dataset ITC2007. Hasilnya SCHC menunjukkan performa yang lebih baik daripada SA dan GD.

Salah satu penelitian lainnya (Cheraitia and Haddadi, 2016) adalah implementasi Simulated Annealing pada ETP dengan dataset Carter (Toronto). Metode ini menggunakan tiga konfigurasi, Exam Shifting, Time Slot Interchange, dan Kempe Chain. Hasil penelitian ini dibandingkan dengan beberapa penelitian terdahulu dan memiliki hasil yang mampu bersaing. Pada peneletian review tentang pendekatan hyper-heuristic pada permasalahan penjadwalan ujian (Peter, 2011), menyebutkan runner-up dari kompetisi ITC 2007 kedua (Gogos, 2012) menjelaskan tahap pengembangan yang dilakukan setelah melakukan inisialisasi solusi, dimana simulated annealing diterapkan agar tidak terjebak dari *local optimum* yang disebabkan karena implementasi dari metode Hill Climbing yang digunakan untuk eksploitasi solusi. Pada penelitiannya, inisialisasi solusi membutuhkan waktu sekitar dua puluh persen dan sisanya digunakan untuk tahap pengembangan.

Penelitian selanjutnya membahas tentang pengurutan pada ujian (Mandal and Kahar, 2015a). Penelitian ini mengurutkan ujian berdasarkan graph heuristic dan dan pada awalnya hanya beberapa ujian yang dijadwalkan dan kemudian dikembangkan menggunakan algoritma GD. Proses ini akan dilanjutkan hingga semua ujian dijadwalkan. Urutan dari penelitian ini adalah Largest Degree (LD), Larget Weighted Degree (LWD), dan Largest Enrollment (LE) dan pengurutan tersebut akan berubah tergantung dari Saturation Degree (SD). Penelitian ini diuji

coba pada dataset Carter dan hasilnya mampu mengungguli metode seperti Hybrid Fish Swarm Optimisation. Sedangkan pada penelitian dengan menggunakan cara yang sama (berbeda urutan), peneliti menggunakan metode Hill Climbing (HC) (Mandal and Kahar, 2015b). Hasilnya metode tersebut lebih bagus daripada metode-metode sebelumnya yang menggunakan HC dengan dataset Carter. Peneliti tidak membandingkan hasil penelitiannya dengan yang sebelumnya, akan tetapi GD memiliki hasil yang lebih bagus daripada HC.

Terdapat penelitian yang mengembangkan pemodelan baru mengenai ETP, yang sebelumnya hanya menggunakan single obyektif dikembangkan menjadi multi obyektif. Penelitian ini (Muklason, 2017) membahas tentang keadilan bagi para pelajar yang dilibatkan dalam ETP. Keadilan tersebut berupa pertimbangan terkait waktu yang diberikan pelajar setelah menghadapi ujian. Hal ini menyebabkan metode yang dikembangkan menjadi multi-obyektif. Algoritma yang digunakan pada strategi seleksi LLH adalah Self-Adaptive Learning, sedangkan untuk move acceptance menggunakan algoritma Great Deluge. Kemudian diujikan pada dataset ITC 2007, Carter, Yeditepe, dan Nott dengan implementasi pada framework Hyflex. Dan hasilnya menunjukkan bahwa metode yang dikembangkan mampu bersaing dengan metode-metode sebelumnya.

Pada penelitian ini (Bahman et al, 2015) mengusulkan dan menganalisa seleksi meta-heuristic dengan strategi seleksi berdasarkan prey-predator untuk mengetahui strategi menawar pada Generating Companies (GENCOs). Peneliti mengusulkan metode prey-predator untuk memecahkan permasalahan tingkatan pertama di bilevel problem dan menggunakan metode iterasi uuntuk mengetahui supply function equilibrium yang optimal pada BS (Bidding Strategies) GENCOs. Penelitian ini mengusulkan metode baru yang dinamakan Improvement of the Original PPOA (IPPO), dimana PPOA merupakan Prey-Predator Optimization Algorithm. Metode IPPO menggunakan teknik metode baru untuk meningkatkan diversity pada saat initial solution untuk meningkatkan kemampuan exploration pada algoritma. Untuk mengevaluasi dan menganalisi pendekatan, peneliti membandingkan dengan beberapa metode PSO (Particle Swarm Optimization) dan PPOA. Hasilnya IPPO berhasil unggul dari PSO dan PPOA pada hampir semua

permasalahan. Hal ini membuktikan bahwa dengan merubah initial solution pada metode dapat merubah performa dari metode tersebut.

Pada penelitiaan (Tilahun, 2013) mengusulkan dan menganalisa seleksi meta-heuristic dengan strategi seleksi berdasarkan metode prey-predator untuk digunakan pada RBFNNs (Radial Basis Function Neural Networks). Penelitian akan terlebih dahulu melakukan training untuk mencari parameternya. Untuk mengevaluasi metode tersebut, maka peneliti membandingkan metodenya dengan GA (Genetic Algorithm). Hasilnya menunjukkan bahwa PPA menunjukkan hasil yang lebih baik dari GA dalam hal meminimalisir kesalahan dan kecepatan.

Pada penelitian (Nawaf, 2018) mengusulkan dan menganalisa seleksi meta-heuristic dengan strategi seleksi berdasarkan metode prey-predator untuk digunakan pada optimasi microchannel heat sinks. Pada penelitian ini metode prey-predator digunakan untuk menemukan parameter yang akan digunakan, seperti thermal resistance dan pumping power of heat sink.

Berdasarkan penelitian (Tilahun, 2017), metode HPPA (Hyper-heuristic Prey Predator Algorithm) yang telah diujikan di dataset CEC2005, CEC2013, CEC2014 telah unggul dibandingkan metode Genetic Algorithm (GA), Particle Swarm Optimization (PSO) dan Simulated Annealing (SA). Metode HPPA ini berfungsi untuk mencari parameter dari PPA sendiri, yaitu jumlah exploration dan exploitation. Pada penelitian (Tilahun, 2019) menggunakan metode meta-heuristic Prey-Predator pada Examination Timetabling dengan dataset Carter. Hasil dari penelitian ini mampu bersaing dengan penelitian sebelumnya.

Berdasarkan state-of-the-art yang telah disebutkan, maka Tabel 2.1 menunjukkan penelitian terdahulu yang telah dilakukan dan memiliki relevansi dengan penelitian ini.

*Tabel 2. 1 Rangkuman Penelitian Terdahulu*

| **Peneliti** | **Dataset** | **Metode** | **Deskripsi** | **Gap** |
|---|---|---|---|---|
| Asaju, 2015 | Carter | Hyper-heuristic, Hybrid Artificial Bee Colony (HABC) | Mengembangkan ABC (Artificial Bee Colony) menjadi HABC, dimana ABC dikembangkan agar tidak terjebak dalam local optimum | Berbeda metode penelitian, Asaju menggunakan ABC, sedangkan penelitian ini menggunakan Prey-Predator. |
| Bykov and Petrovic, 2016 | ITC 2007 | Step Counting Hill Climbing (SCHC) | Mengembangkan Hill Climbing menjadi Step Counting Hill Climbing dimana iterasi digunakan agar Hill Climbing tidak terjebak kedalam local optimum | Berbeda metode penelitian, pada penelitian tersebut menggunakan SCHC sedangkan penelitian ini menggunakan Prey-Predator |
| Cheraitia and Haddadi, 2016 | Carter | Simulated Annealing | Penelitiian ini mengimplementasi Simulated Annealing pada ETP menggunakan konfigurasi, Exam Shifting, Time Slot Interchange, dan Kempe Chain | Penelitian tersebut berfokus pada low-level heurisitic sedangkan penelitian ini berfokus pada kombinasi seleksi LLH dan move acceptance |
| Gogos, 2012 | ITC 2007 | Simulated Annealing | Penelitian menjelaskan tahap pengembangan yang dilakukan setelah melakukan inisialisasi solusi, dimana simulated annealing diterapkan agar tidak terjebak dari *local optimum* | Penelitian tersebut menggunakan Simulated Annealing agar Hill Climbing tidak terjebak pada local optimum. Sementara penelitian ini |

| | | | | |
|---|---|---|---|---|
| | | | yang disebabkan karena implementasi dari metode Hill Climbing yang digunakan untuk eksploitasi solusi | menggunakan Prey-Predator dengan Simulated Annealing |
| Mandal and Kahar, 2015a | Carter | Great Deluge | Penelitian ini menggunakan graph heuristic untuk inisialisasi solusi, dengan urutan Largest Degree (LD), Larget Weighted Degree (LWD), dan Largest Enrollment (LE) dan pengurutan tersebut akan berubah tergantung dari Saturation Degree (SD) | Penelitian tersebut menggunakan dataset Carter, sedangkan pada penelitian ini akan menggunakan dataset ITC 2007 |
| Muklason, 2017 | Carter, ITC 2007, Yeditepe , Nott | Self Adaptive – Great Deluge | Penelitian membahas tentang keadilan bagi para pelajar yang dilibatkan dalam ETP. Keadilan tersebut berupa pertimbangan terkait waktu yang diberikan pelajar setelah menghadapi ujian. Hal ini menyebabkan metode yang dikembangkan menjadi multi-obyektif | Penelitian tersebut menggunakan metode Self Adaptive – Great Deluge, sedangkan penelitian ini menggunakan metode Prey-Predator – Simulated Annealing |
| Bahman et al, 2015 | | Meta-heuristic Prey-Predator | Penelitian ini menggunakan Prey-Predator sebagai strategi | Penelitian tersebut menggunakan Prey-Predator |

| | | | | |
|---|---|---|---|---|
| | | | menawar pada Generating Companies (GENCOs) | sebagai strategi tawar menawar, sedangkan pada penelitian ini Prey-Predator digunakan untuk mencari solusi pada Examination Timetabling |
| Tilahun, 2013 | | Meta-heuristic Prey-Predator | Penelitian tersebut menggunakan metode pada RBFNNs (Radial Basis Function Neural Networks) | Penelitian tersebut menggunakan meta-heurisitc Prey-Predator pada RBFNNs, sedangkan penelitian ini menggunakan hyper-heuristic Prey-Predator pada Examination Timetabling |
| Nawaf, 2018 | | Meta-heuristic Prey-Predator | Penelitian tersebut menggunakan Prey-Predator sebagai optimasi *microchannel heat sinks* | Penelitian tersebut digunakan untuk optimasi *microchannel heat sinks,* sedangkan penelitian ini digunakan untuk menemukan solusi pada Examination Timetabling |
| Tilahun, 2017 | | Hyper-Heuristic Prey-Predator | Penelitian ini mengusulkan Prey-Predator yang bermula meta-heuristic menjadi hyper-heuristic dan digunakan pada | Penelitian akan menggabungkan Prey-Predator dengan Simulated Annealing dan akan digunakan untuk mencari |

| | | | | |
|---|---|---|---|---|
| | | | sejumlah dataset CEC2005, CEC2013, CEC2014 | solusi pada Examination Timetabling |
| Tilahun, 2019 | Carter | Meta-heuristic Prey-Predator | Penelitian ini menggunakan menggunakan metode untuk mencari solusi Examination Timetabling pada dataset Carter | Penelitian ini menggunakan hyper-heurisitic Prey-Predator dan Simulated Annealing dan akan digunakan untuk mencari solusi pada permasalahan Examination Timetabling dengan dataset ITC 2007 |

Dari state-of-the-art yang telah disebutkan sebelumnya, maka muncul ide untuk menyelesaikan permasalahan penjadwalan ujian (ETP) dengan menggunakan pendekatan hyper-heuristic yang dioptimasi dengan strategi seleksi algoritma Hyper-heuristic Prey Predator Algorithm (HPPA) serta menggunakan move acceptance algoritma Simulated Annealing (SA). Hal ini yang membedakan penelitian kami dengan peneliti terdahulu, sekaligus diharapkan mampu memberikan improvement pada framework Hyflex terkini.

## 2.2 Dasar Teori

Penelitian ini akan mengevaluasi performa dari algoritma yang digunakan untuk seleksi heuristik dalam penyelesaian permasalahan penjadwalan ujian. Algoritma ini akan diuji coba pada dataset ITC 2007 untuk menemukan solusi optimum (atau hampir optimum). ITC 2007 berisi “masalah penjadwalan ujian” biasa disebut permasalahan penjadwalan ujian (*Examination Timetabling Problem*). Berikut ini adalah dasar dari teori yang digunakan pada penelitian ini.

### *2.2.1 Educational Timetabling Problem*

Permasalahan *timetabling* dapat didefinisikan sebagai permasalahan meletakkan sejumlah *events* ke dalam periode waktu dan ruang yang terbatas. 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. Permasalahan *timetabling* telah diaplikasikan pada bidang pendidikan, olahraga, transportasi, karyawan. Jenis penjadwalan pada bidang pendidikan yaitu *examination timetabling* dan *course timetabling* (Gaspero et al., 2007). Perbedaan *examination timetabling* dengan *course timetabling* adalah pada ruangan dan periode waktu (Burke and Petrovic, 2002). 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 (Ilyas and Iqbal, 2015) :

- 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 (Burke and Petrovic, 2002) :

- 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 Exam Timetabling Problem

Permasalahan penjadwalan ujian adalah masalah yang populer di dunia akademis yang berkaitan dengan pengalokasian ujian ke sejumlah slot waktu terbatas (periode) yang memiliki batasan (*constraint*) sehingga tidak ada siswa memiliki dua atau lebih ujian pada waktu yang sama dan pengalokasian ujian tidak boleh melebihi kapasitas ruangan (Turabieh and Abdullah, 2011). Berikut ini adalah *hard constraint* yang secara umum telah dirangkum dari penelitian:

1. Tidak ada ujian dengan *resource* (misalnya: siswa) yang ditetapkan secara bersamaan; dan
2. *Resource* harus cukup (misalnya: jumlah siswa yang ditempatkan ke suatu ruangan harus di bawah kapasitas ruangan. Dengan kata lain, adanya ruangan yang cukup untuk semua ujian)

Adapun *soft constraint* yang secara umum, yaitu:

1. Tempatkan ujian-ujian yang berkonflik di hari yang berbeda, tidak dalam slot waktu yang sama atau hari yang berturut-turut;
2. Sekelompok ujian harus berlangsung pada waktu yang sama, pada hari yang sama atau di satu lokasi;
3. Ujian dijadwalkan secara berurut;
4. Jadwalkan semua ujian, atau ujian dengan jumlah peserta terbanyak, secepat mungkin;
5. Kebutuhan tertentu dari ujian harus dipenuhi;
6. Jumlah siswa atau ujian dibatasi dalam setiap timeslot;
7. Waktu ujian yang dibutuhkan;
8. Ujian yang berkonflik pada hari yang sama dialokasikan pada jadwal yang berdekatan;
9. Ujian dapat dialokasikan pada ruangan yang sama;
10. Hanya ujian dengan panjang waktu ujian yang sama dapat dialokasikan ke ruangan yang sama; dan
11. Kebutuhan *resource* (misalnya: fasilitas ruang ujian).

### 2.2.3 Dataset ITC 2007 (Examination Timetabling Problem)

Pada umumnya, dataset dapat diartikan sebagai sekumpulan data dimana setiap kolom tabel mewakili variabel tertentu. Variabel tersebut kemudian dibuat semirip mungkin dengan skenario yang ada di dunia nyata agar dapat digunakan untuk pengujian dalam mencari solusi dari sebuah masalah (Sabar et al., 2012). Perkuliahan, ujian, kendaraan atau transportasi, hingga pekerjaan seseorang adalah beberapa contoh entitas yang variabelnya dapat direpresentasikan dalam sebuah dataset. Selama beberapa dekade terakhir, terdapat sebuah dataset yang menarik perhatian para peneliti yaitu dataset penjadwalan khususnya di lingkup pendidikan. Penelitian ilmiah yang pertama kali membahas masalah penjadwalan di mulai sejak tahun 1960 hingga menjadi salah satu penelitian populer saat ini (Dornelles, 2015).

Terdapat kategori penting pada Permasalahan penjadwalan ujian yang oleh peneliti akan dibagi menjadi tiga, yaitu:

**1) *Decision Variable***

Berikut adalah variabel-variabel yang akan digunakan dalam dataset ITC 2007, deskripsinya bisa dilihat pada tabel 2.1 di bawah ini.

*Tabel 2. 2 Instance Dataset ITC 2007*

| ***Problem Instances*** | ***Time Slots*** | ***Exams*** | ***Students*** | ***Rooms*** | ***Conflict Density*** |
|---|---|---|---|---|---|
| EXAM1 | 54 | 607 | 7981 | 7 | 0.05 |
| EXAM2 | 40 | 870 | 12743 | 49 | 0.01 |
| EXAM3 | 36 | 934 | 16439 | 48 | 0.03 |
| EXAM4 | 21 | 273 | 5045 | 1 | 0.15 |
| EXAM5 | 42 | 1018 | 9253 | 3 | 0.01 |
| EXAM6 | 16 | 242 | 7909 | 8 | 0.06 |
| EXAM7 | 80 | 1096 | 14676 | 15 | 0.02 |
| EXAM8 | 80 | 598 | 7718 | 8 | 0.05 |
| EXAM9 | 25 | 169 | 655 | 3 | 0.08 |
| EXAM10 | 32 | 214 | 1577 | 48 | 0.05 |
| EXAM11 | 26 | 934 | 16439 | 40 | 0.03 |
| EXAM12 | 12 | 78 | 1653 | 1 | 0.18 |

Sebagaimana yang ditampilkan pada tabel 2.1, problem *instance* dari dataset ITC 2007 memiliki minimal 78 ujian yang harus dialokasikan pada slot waktu dan ruang ujian. Ini berarti, problem instance dari dataset ITC 2007 memiliki *decision variable* minimal 156 (Muklason, 2017).Sedangkan problem *instance* yang memiliki 1096 ujian sebagai nilai tertinggi dari dataset ITC 2007 berarti terdapat maksimal 2192 *decision variable*. Representasinya seperti berikut:

$$X_{ip}^{p} \begin{cases} 1 & \text{Jika ujian i pada time slot p} \\ 0 & \text{Jika tidak} \end{cases} \quad (2.1)$$

$$X_{ir}^{R} \begin{cases} 1 & \text{Jika ujian i pada ruangan r} \\ 0 & \text{Jika tidak} \end{cases} \quad (2.2)$$

**2) *Hard Constraint* dan *Soft Constraint***

Dataset ITC 2007 dikenalkan sebagai bentuk pengembangan Toronto yang merupakan salah satu *benchmark* dataset. Perbedaannya terletak pada adanya penambahan *hard* dan *soft constraint* yang baru yang lebih kompleks (McCollum et al., 2012). Untuk memudahkan pembacaan formula, representasi set dan parameter yang berhubungan dengan berbagai formulasi diuraikan sebagai berikut:

1) Jika berhubungan dengan formula Ujian adalah sebagai berikut:

E = set ujian;

$S_i^E$ = size ujian i ∈E ;

$d_i^E$ = durasi waktu ujian i ∈E ;

$f_i^E$ = sebuah boolean yang akan bernilai 1 jika ujian i adalah subjek dari pinalti dari front load;

D = set dari durasi waktu yang digunakan $\bigcup_i d_i^E$ ; dan

$u_{id}^D$ = sebuah boolean bernilai 1 jika dan hanya jika ujian memiliki "durasi" d ∈ D

2) Jika berhubungan dengan formula Siswa adalah sebagai berikut:

S = set siswa

$$t_{is} \begin{cases} 1 & \text{Jika siswa s mengikuti ujian i} \\ 0 & \text{Jika tidak} \end{cases} \qquad (2.3)$$

3) Jika berhubungan dengan formula Ruang ujian adalah sebagai berikut:

R = set ruang ujian

$s_r^R$ = size ruang ujian r ∈ R

$w_r^R$ = bobot yang menentukan penalti yang akan digunakan ruangan r

4) Jika berhubungan dengan formulasi Periode Waktu adalah sebagai berikut:

$d_p^P$ = durasi periode waktu p ∈ P

$f_p^P$ = sebuah Boolean yang akan bernilai 1 jika periode waktu p adalah subjek dari penalti Front Load, jika tidak maka 0. Parameter kedua dari *Front Load*.

$f_p^P$ = 1

$w_p^P$ = bobot yang menentukan penalti yang akan digunakan oleh periode p

$y_{pq}$ = sebuah Boolean yang akan bernilai 1 jika periode waktu p dan q terdapat di hari yang sama, jika tidak maka 0.

Berikut ini adalah *hard constraint* yang terdapat pada ETP dari dataset ITC 2007 :

1. *No Conflicts* $\forall i \in E$ - ujian yang bertentangan atau memiliki konflik tidak dapat dijadwalkan pada slot waktu yang sama, setiap ujian diselenggarakan hanya pada satu ruangan

$$\sum_{r \in R} X_{ir}^R \geq 1 \tag{2.4}$$

dan satu periode waktu

$$\sum_{p \in P} X_{ip}^P \geq 1 \tag{2.5}$$

2. *Room occupancy* – untuk setiap ruang $\forall r \in R$ dan waktu ujian $\forall p \in P$, tidak ada tambahan penggunaan meja atau kursi. Gunakan meja atau kursi yang tersedia pada ruangan tersebut

$$\sum_{i \in E} s_i^E \, X_{ip}^P X_{ir}^R \leq s_r^R \tag{2.6}$$

3. *Period utilization or Period duration* – ujian $\forall i \in E$ tidak dilaksanakan lebih dari batas waktu $\forall p \in P$ yang telah ditetapkan $d_i^E X_{ip}^P \leq d_p^p$

*e*

$$d_i^E X_{ip}^P \leq d_p^p \tag{2.7}$$

4. *Period related* - Satu set persyaratan pemesanan waktu antara ujian satu dan lainnya (contoh: ujian i dan j) harus dipenuhi. Berikut adalah beberapa constraint yang lebih spesifik khusus untuk penjadwalan ujian yang memiliki persyaratan:

a. *After constraint* $\forall(i,j) \in H^{\text{aft}}$ - i harus segera dilaksanakan setelah j

$$X_{ip}^{P} + X_{jq}^{P} \leq 1; \forall(p,q) \in P \quad (2.8)$$

b. *Exam coincidence* $\forall(i,j) \in H^{\text{coin}}$ - i harus dilaksanakan pada waktu yang sama dengan j

$$X_{ip}^{P} = X_{jp}^{P}; \forall p \in P \quad (2.9)$$

c. *Period Exclusion* $\forall(i,j) \in H^{\text{excl}}$ - i tidak dapat dilaksanakan pada waktu yang sama dengan j

$$X_{ip}^{P} + X_{jp}^{P} \leq 1; \forall p \in P \quad (2.10)$$

5. *Room related* – berikut ini persyaratan ruang ujian khusus untuk ujian tertentu:

a. *Room exclusive* $\forall(i,j) \in \text{H}^{sole}$ - ujian i harus atau hanya dapat dilaksanakan di ruangan tertentu

$$X_{ip}^{P} + X_{ir}^{R} + X_{jp}^{P} + X_{jr}^{R} \leq 1; \forall j \in E; j \neq i; \forall p \in P; \forall r \in R \quad (2.11)$$

*Berikut adalah Soft constraint* dari dataset ITC 2007 :

1. *Two exams in a row* ($C_s^{2R}$) – penalti yang terjadi setiap kali seorang siswa harus mengikuti dua ujian berbeda yang dijadwalkan dalam dua slot waktu berturut-turut dalam hari yang sama

$$C^{2R} = \sum_{a \in A} w_a^c C_a^{2R} \quad (2.12)$$

$$X_{ip}^{P} + X_{jq}^{P} \leq 1 + C_a^{2R} \quad (2.13)$$

$$\forall a = (i,j) \in A; \forall (p,q) \in P; |q-p| = 1; Y_{pq} = 1 \quad (2.14)$$

dimana:

a = gabungan kedua ujian i dan j

A = jumlah atau banyaknya siswa yang mengikuti dua ujian berbeda yang dijadwalkan dalam dua slot waktu berturut-turut dalam hari yang sama.

2. *Two exams in a day ($C_s^{2D}$)* – penalti yang terjadi setiap kali seorang siswa harus mengikuti dua ujian berbeda yang dijadwalkan dalam dua slot waktu yang tidak berurutan dalam hari yang sama

$$C^{2R} = \sum_{a \in A} w_a^c C_a^{2D} \quad (2.15)$$

$$X_{ip}^P + X_{jq}^P \leq 1 + C_a^{2R} \quad (2.16)$$

$$\forall a = (i,j) \in A; \forall (p,q) \in P; |q-p| \geq 2; Y_{pq} = 1 \quad (2.17)$$

3. *Period spread ($C_S^{pS}$)* – penalti yang terjadi setiap kali seorang siswa memiliki lebih dari satu ujian dalam jangka waktu tertentu

$$C^{PS} = \sum_{a \in A} w_a^c C_a^{PS} \quad (2.18)$$

$$X_{ip}^P + X_{jq}^P \leq 1 + C_a^{PS} \quad (2.19)$$

$$\forall a = (i,j) \in A; \forall (p,q) \in P; 1 \leq |q-p| \leq g \quad (2.20)$$

dimana:

g = *gap period* antara ujian i dan j

4. *No Mixed durations ($C_{pr}^{NMD}$)* – penalti yang terjadi setiap kali ada beberapa ujian menggunakan ruang ujian yang sama tetapi dengan periode waktu yang berbeda

$$C^{NMD} = \sum_{p \in P} \sum_{a \in A} C_{pr}^{NMD} \tag{2.21}$$

dimana:

pr = sepasang periode waktu dan ruangan

5. *Front load ($C^{FL}$)* – penalti yang terjadi ketika menjadwalkan ujian dengan resource besar $f_i^E = 1$ di akhir periode ujian$f_p^P = 1$

$$C^{FL} = \sum_{i \in E} \sum_{p \in P} f_i^E f_p^P X_{ip}^P \tag{2.22}$$

6. *Room ($C^R$)* dan *period penalties ($C^P$)* –$C^R$adalah penalti yang berhubungan dengan memberikan ijin penggunaan ruang ujian kapanpun waktu penyelenggaraannya.$C^P$ adalah penalti yang berhubungan dengan alokasi waktu kapanpun untuk diselenggarakan sebuah ujian.

$$C^R = \sum_{r \in R} \sum_{i \in E} w_r^R X_{ir}^R \tag{2.23}$$

$$C^P = \sum_{p \in P} \sum_{i \in E} w_p^P X_{ip}^P \tag{2.24}$$

3) *Objective Function*

Setiap dataset memiliki bobot penalti masing-masing (Abdullah and Alzaqebah, 2013). Sehingga objective function dari dataset ITC 2007 berfungsi untuk meminimalkan nilai bobot penalti (W) dengan persamaan sebagai berikut:

$$\sum_{s\in S}(w^{2R}C_s^{2R} + w^{2D}C_s^{2D} + w^{PS}C_s^{PS}) + w^{NMD}C^{NMD} + w^{FL}C^{FL} + C^P + C^R \quad (2.25)$$

dimana:

$w^{2R}$: bobot untuk *two in a row*$C_s^{2R}$

$w^{2D}$: bobot untuk *two in a day*$C_s^{2D}$

$w^{PS}$: bobot untuk *period spread*$C_s^{PS}$

$w^{NMD}$: bobot untuk *No mixed duration*$C_{pr}^{NMD}$

$w^{FL}$: bobot untuk penalti *Front Load*$C^{FL}$

Pada tabel 2.3 diperlihatkan nilai pelanggaran dari dataset ITC 2007 pada permasalahan examination timetabling.

*Tabel 2. 3 Bobot Pelanggaran dataset ITC 2007 examination timetabling*

| EXAM | $W^{2D}$ | $W^{2R}$ | $W^{PS}$ | $W^{NMD}$ | $W^{FL}$ |
|---|---|---|---|---|---|
| **1** | 5 | 7 | 5 | 10 | 5 |
| **2** | 5 | 15 | 1 | 25 | 5 |
| **3** | 10 | 15 | 4 | 20 | 10 |
| **4** | 5 | 9 | 2 | 10 | 5 |
| **5** | 15 | 40 | 5 | 0 | 10 |
| **6** | 5 | 20 | 20 | 25 | 15 |
| **7** | 5 | 25 | 10 | 15 | 10 |
| **8** | 0 | 150 | 15 | 25 | 5 |

### 2.2.4 Hyper-heuristics (HH)

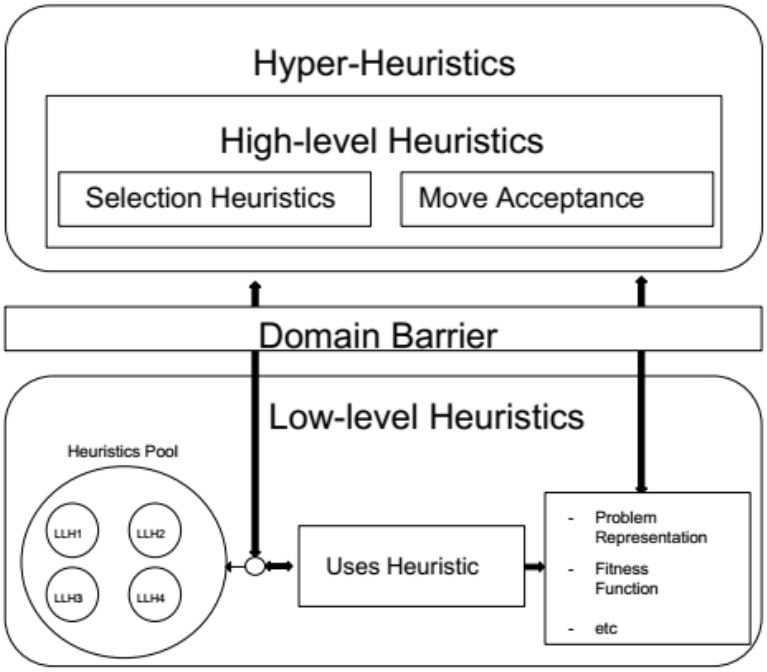
*Hyper-heuristics* merupakan bidang penelitian penting pada area optimasi, dapat didefinisikan sebagai metode pencarian yang bertujuan untuk menyelesaikan permasalahan optimasi dengan memilih atau menghasilkan heuristik (Burke et al., 2010).

Teorema tidak ada makan siang gratis menetapkan bahwa "untuk algoritma apa pun, setiap peningkatan kinerja atas satu kelas masalah diimbangi oleh kinerja yang berkurang di kelas lain" (Cowling et al., 2000). Teorema ini berlaku untuk Metaheuristik, karena untuk menemukan solusi yang baik, metode-metode tersebut seringkali perlu dirancang dan disetel ke beberapa problem domain atau bahkan

hanya single problem. *Hyper-heuristics* mengatasi hal ini dengan meningkatkan *generality*.

*Hyper-heuristics* membagi heuristik menjadi 2 tipe, yaitu *high-level* dan *low-level*. Pada dasarnya *high-level* heuristik bertanggung jawab untuk memilih *low-level* heuristik yang akan diterapkan dan solusi mana yang akan diterima untuk menggantikan solusi yang kurang optimum. Sedangkan *low-level* heuristik bertanggung jawab dalam memecahkan masalah atau mencari ruang solusi. *High-level* heuristik bersifat independen, sedangkan *low-level* heuristik masih bergantung pada permasalahan.

*Hyper-heuristic* (HH) adalah pencarian tingkat tinggi, selanjutnya akan disebut *high level* heuristik, dijalankan untuk mendukung algoritma tradisional dari heuristik. Istilah HH pada dasarnya adalah sebuah ide untuk mengembangkan lebih banyak algoritma yang dapat digunakan secara umum (Muklason, 2017). Idenya dapat dilakukan dengan mengembangkan kelebihan dari setiap algoritma pada LLH atau mengkombinasikan dua atau lebih algoritma. Framework dasar dari HH dapat dilihat pada Gambar 2.1. Bagian inti dari framework HH adalah *domain barrier*. Bagian tersebut memisahkan HH dari problem domain, sehingga yang HH dapat langsung memilih informasi dan LLH mana yang dapat dipilih untuk menyelesaikan masalah apa. Tidak seperti algoritma LLH, pendekatan ini berfokus pada pencarian ruang heuristik daripada pencarian solusi. Selain itu, HH menggunakan LLH untuk memberikan solusi berbagai masalah daripada memberikan solusi khusus untuk masalah tertentu, dapat menangani berbagai contoh masalah dengan karakteristik yang berbeda tanpa memerlukan intervensi ahli.

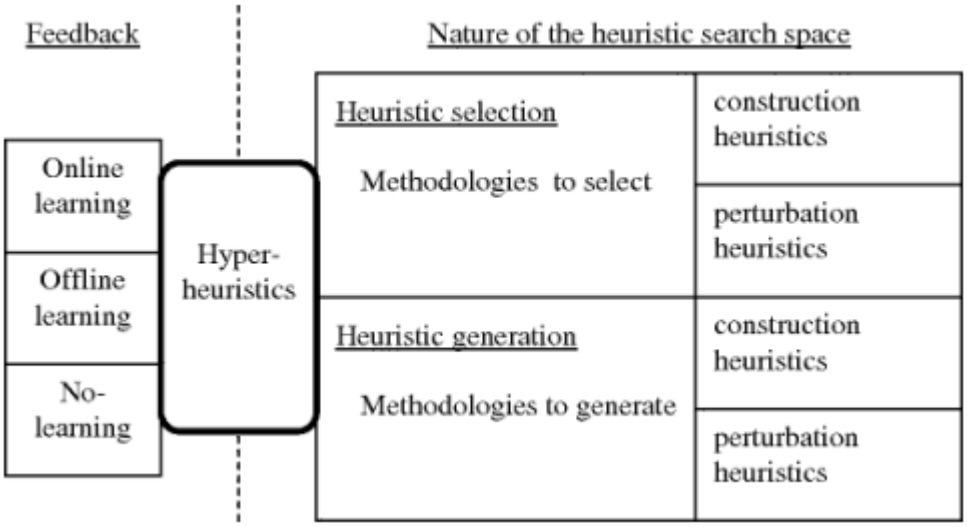



*Gambar 2. 1 Framework Hyper-heuristic (Ferreira, 2016)*

Berdasarkan Gambar 2.2, pendekatan HH diklasifikasikan dalam dua dimensi yaitu: 1) klasifikasi berdasarkan mekanisme *feedback/learning*; dan 2) *search space* heuristik pada umumnya. Pada kategori pertama, pendekatan HH dianggap sebagai *online learning* ketika membutuhkan *feedback* di dalam proses pencarian, sebaliknya pada *no learning*. Sedangkan pendekatan *offline learning* digunakan pada HH ketika ide untuk proses pencarian diperoleh dari hasil pengujian terhadap *instance* training data secara umum sehingga solusi akan digeneralisasikan.

Pada kategori kedua, masing-masing *search space* pada level pertama dikategorikan menjadi heuristik *selection* dan heuristik *generation*, selanjutnya terbagi lagi dalam dua metodologi yaitu *construction* heuristik dan *perturbation* heuristik pada level kedua (Muklason, 2017). Framework HH telah menyediakan set spesifik masalah dan tantangan dari pendekatan *construction* heuristik adalah terhadap solusi yang dibangun secara bertahap untuk mencapai solusi lengkap dengan memilih heuristik yang paling atau hampir cocok untuk menyelesaikan masalah tersebut. Urutan heuristik yang akan dipilih ditentukan oleh ukuran

masalah combinatorial yang akan diselesaikan. Proses selesai ketika solusi lengkap telah dicapai.

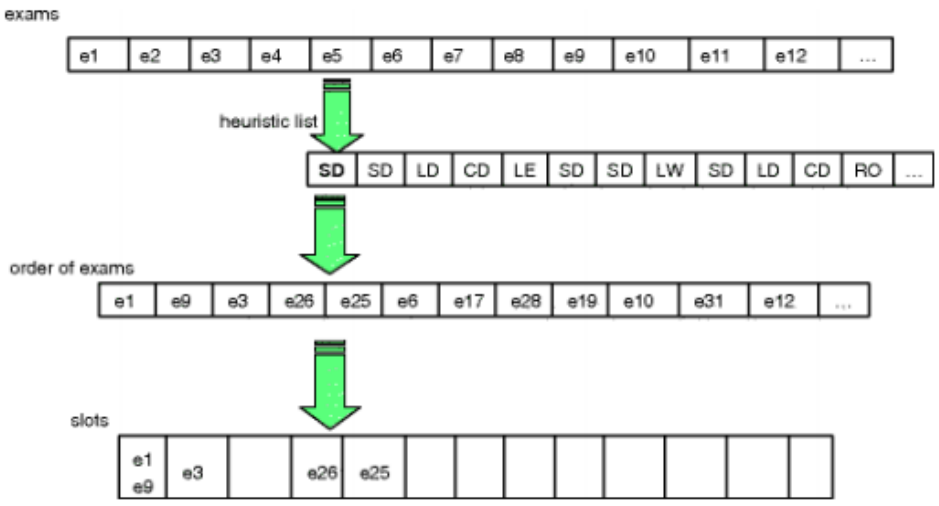



*Gambar 2. 2 Klasifikasi Hyper-heuristic (Burke et la., 2010)*

Beberapa survei mengenai *state-of-the-art* dari formulasi Permasalahan penjadwalan ujian dimulai dari algoritma tradisional *graph colouring*, hingga metaheuristik dan *hyper-heuristic* telah dilakukan oleh beberapa peneliti (Muklason, 2017). Gambar 2.3 merupakan salah satu contoh yang mewakili *online learning* pada *construction* heuristik menggunakan *graph-colouring* pada *examination timetable problem*, dimana graph heuristik menggunakan informasi untuk mengurutkan *event* (ujian) berdasarkan karakteristiknya. Karakteristik yang dimaksud dapat berupa jumlah konflik dengan event lain atau level konflik event tersebut. Berikut ini adalah penjelasan lima heuristik *graph-colouring* (berdasarkan Gambar 2.3):

1. *Largest degree* (LD): Mengurutkan event dari besar ke kecil (*decrease*) secara bertahap berdasarkan besarnya konflik dengan *event* lain
2. *Largest weighted degree* (LWD): Sama seperti LD tetapi urutan berdasarkan pembobotan *event* dari banyaknya siswa yang terlibat atau banyaknya peserta ujian

3. *Colour degree* (CD): Mengurutkan *event* dari besar ke kecil (*decrease*) dengan ketentuan jumlah dari setiap konflik pada *event* (*event* dengan keterlibatan siswa yang sama) dijadwalkan pada slot waktu yang telah diisi oleh *event* lainnya.
4. *Largest enrolment* (LE): Mengurutkan event dari besar ke kecil (*decrease*) secara bertahap berdasarkan jumlah pendaftaran
5. *Saturation degree* (SD): Mengurutkan event dari besar ke kecil (*decrease*) secara bertahap berdasarkan jumlah timeslot yang tersedia untuk setiap *event* yang terdaftar pada saat itu




*Gambar 2. 3 Online Learning Heuristik Graph-Colouring Timetable*

Solusi dicapai dengan mempertimbangkan secara iteratif daftar heuristik yang akan digunakan untuk mengurutkan *event* yang belum terjadwal. Salah satu contoh yang mewakili *offline learning* berdasarkan *construction* heuristik yaitu pada masalah pengklasifikasian sistem *bin packing.*(Burke et al., 2010), menuliskan salah satu penelitian yang mengangkat pengklasifikasian sistem *bin packing* atau sistem pengangkutan tempat sampah sebagai problem domain memilih empat heuristik*bin packing* dari literatur. Pemilihan dilakukan berdasarkan hasil terbaik yang diperoleh pada pengujian terhadap *benchmark* dataset. Kedua contoh tersebut, *online* dan *offline learning*, selengkapnya dapat dilihat pada penelitian (Burke et al., 2010).

Disatu sisi, pendekatan berdasarkan *perturbation* heuristik tidak memiliki kondisi dasar untuk berhenti, berbeda dengan pendekatan sebelumnya. Pendekatan ini akan mulai dari solusi yang ada sebelumnya, dibentuk secara random maupun menggunakan *constructive* heuristik yang sederhana dapat dilihat pada tabel 2.2. Setelah itu dilakukan peningkatan secara iteratif terhadap solusi tersebut. (Burke et al., 2010) menuliskan bahwa *perturbation* hyper-heuristic berdasarkan beberapa penelitian, dibagi dalam dua proses yaitu: 1) Pemilihan LLH; dan 2) Strategi move acceptance.

*Tabel 2. 4* Construction Heuristik Approach (Burke et al., 2010)

| Problem Domain | Online Learning | Offline Learning |
|---|---|---|
| 1. Bin Packing<br>2. Timetabling<br>3. Penjadwalan Produksi<br>4. Stock Cutting | Pendekatan Metaheuristik:<br>1. Algoritma Genetik<br>2. Random Search<br>3. Dan strategi pencarian single-point lainnya | 1. Learning Classifier Systems<br>2. Algoritma Messy Genetik<br>3. Case-based Reasoning |

LLH selection adalah strategi dimana hyper-heuristic (HH) sebagai *high level* heuristik dapat memilih atau menolak LLH yang ada untuk menyelesaikan problem domain. Sedangkan MA merupakan komponen penting pada pencarian lokal heuristik. Strategi MA terbagi menjadi dua kategori, yaitu: *deterministic* dan *non deterministic*. Istilah lainnya adalah kategori all *move accepted* dan *only improving move accepted*. Diakatakan kategori *deterministic* jika strategi membentuk hasil yang sama dengan kandidat solusi yang digunakan pada saat pengujian MA. Namun, ketika kandidat solusi melibatkan parameter lain, misalnya parameter waktu saat ini, maka dapat dikatakan bahwa strategi yang digunakan merujuk pada kategori *non deterministic* (Burke et al., 2010).

### 2.2.5 *Prey Predator Algorithm* (*PPA)*

Algoritma meta-heuristic yang terinspirasi dari hubungan predator dan mangsanya (Tilahun and Ong, 2015). Didalam algoritma akan membentuk solusi awal secara acak yang kemudian akan dikategorikan menjadi tiga, berdasarkan

performa *objective function* yang telah ditentukan. Solusi dengan performa terbaik dinamakan *best prey* ($X_b$), solusi dengan performa terburuk dinamakan *predator* ($X_p$) dan sisanya *prey* biasa. Predator bekerja sebagai agen untuk melakukan exploration di solution space dengan cara menakuti prey dan secara acak berlarian. Oleh karena itu, updatenya dilakukan dengan cara berlari mengejar prey yang lemah (prey yang memiliki performa paling rendah). Para prey akan melarikan diri dari predator dan mencoba mengikuti best prey, yang memiliki nilai kelangsungan hidup yang besar. Untuk mengupdate predator sendiri dapat dilihat pada persamaan 2.26.

$$X_p := X_p + \lambda_{max}\, \mathrm{U}_r + (\lambda_{min}\mathrm{rand})\mathrm{U}_c \tag{2.26}$$

. Dimana $\lambda_{min}$ dan $\lambda_{max}$ merupakan parameter algoritma untuk menentukan maksimum panjang langkah exploration dan exploitation, $U_r$ adalah random direction, rand adalah nomor acak antara 0 dan 1 dan Uc bisa dilihat pada persamaan 2.27

$$Uc = \begin{cases} \frac{x^i - x_p}{\|x^i - x_p\|}, & jika\ x^i \neq x_p \\ 0, & \end{cases} \tag{2.27}$$

Pembaruan best prey dilakukan menggunakan local search. Agar dapat melakukan search, maka m random akan dibuat, dimana m merupakan parameter algoritma, dan update akan dilakukan di *direction* yang dapat meningkatkan performa best prey. Jika tidak terdapat *direction* diantaranya, maka m akan berada pada posisi semula. Rumus update pada best prey bisa dilihat pada persamaan 2.28

$$x_b := x_b + \lambda_{min} U_I \tag{2.28}$$

$$Dimana\ U_I = argmin\left\{f(x_b + \lambda_{min}U) | U \in \{U_1, U_2, \dots, U_m, \bar{0}\}\right\}$$

$$dan\ \bar{0}\ adalah\ zero\ factor$$

Untuk Prey biasa $x_i$, yang mana bukan predator ataupun best prey, update akan dilakukan berdasarkan *probability of follow-up*. Probability of follow-up ($P_f$) merupakan parameter algoritma pengontrol antara solusi untuk berlari menjelajahi solution space atau mengikuti best prey sambal melakukan local search. Oleh

karena itu, jika nomor acak sama dengan $P_f$ solusi akan mengikuti prey yang lebih baik dari dirinya sendiri, selain itu akan secara acak melarikan diri dari predator. Hal ini bisa dilihat pada persamaan 2.29

$$x_i := \begin{cases} x_i + (\lambda_{max} rand) U_f + \lambda_{min} U_I, \ jika\ rand \leq P_f \\ x_i + \lambda_{max} U_{run}, \end{cases} \quad (2.29)$$

Dimana $U_{run}$ adalah random direction untuk melarikan diri dari predator, $U_I$ adalah local search direction. Berikut adalah algoritma PPA :

```
1:  Input λmin, λmax, Pf, m, np, nb
2:  Randomly generate N feasible solutions, say {x1, x2,…, xn}
3:  y=sort(x) such that f(yi+1) ≤ f(yi), ∀i
4:     (i.e best prey ={yj|j=N-nb+j for j=1,2,…,nb}
         and the predators={yj|for i=1,2,…,np})
5:  Update predator yi using (2.2) with Xi=ynp+1
6:  Update the best prey yj using (2.3)
7:  For k=np+1:N-nb
8:      Update yk using (2.4)
9:  End for
10: x=y
11: if termination criteria is not met go back to line 3
12: Output yN
```

Di dalam PPA salah satu cara untuk menentukan kadar exploration dan exploitation adalah dengan menyesuaikan probability of follow-up ($P_f$). Jika $P_f$ tinggi kemungkinan prey biasa akan berfokus pada exploitation. Metode kedua adalah dengan menyesuaikan jumlah best prey dan predator. Prey-predator hyper-heuristic (Tilahun, 2017) dibuat berdasarkan metode kedua tersebut, yaitu mengontrol dan menentukan jumlah rasio terbaik dari best prey dan predator. Sebagai contoh jumlah $n_p$ dan jumlah best prey $n_b$ dari jumlah solusi yang ada. T merupakan proporsi dari solusi yang menjadi best prey dan predator, seperti pada persamaan 2.30.

$$T = \frac{n_b + n_p}{N} \quad (2.30)$$

Berhubung semua solusi tidak dapat dijadikan best prey dan predator, maka T<1. B merupakan rasio best prey pada total jumlah predator dan best prey, seperti pada persamaan 2.31. Oleh karena itu, jumlah best prey dan predator $n_p+n_b$ dapat dihitung menggunakan persamaan 2.32. Dimana N merupakan jumlah dari solusi.

Jumlah best prey dapat dihitung menggunakan persamaan 2.33. Jumlah predator otomatis dapat dihitung menggunakan persamaan 2.34.

$$B = \frac{n_b}{TN} \tag{2.31}$$

$$n_p + n_b = n_{pb} = round(TN) \tag{2.32}$$

$$n_b = round(Bn_{pb}) \tag{2.33}$$

$$n_p = n_{pb} - n_b \tag{2.34}$$

$X_{pb}$ dan $X_b$ merupakan jumlah rasio untuk T dan B, contoh untuk T terdapat 1, 2, …, $X_{pb}$ kemungkinan dan B terdapat 1, 2, …, $X_b$. Hal ini dapat dideskripsikan menggunakan matrix $X_{pb}$ dan $X_b$ yang setiap baris berhubungan dengan rasio jumlah predator dan best prey dan setiap kolom merupakan rasio dari jumlah best prey terhadap TN. Hal pertama yang harus dilakukan adalah mendefinisikan matrix untuk menghitung probabilitas pemilihan konfigurasi yaitu LM(t). Matrix tersebut dapat dilihat di Gambar 2.4.

$$LM(t) = \begin{pmatrix} L_{1,1}(t) & L_{1,2}(t) & L_{1,3}(t) \\ L_{2,1}(t) & L_{2,2}(t) & L_{2,3}(t) \\ L_{3,1}(t) & L_{3,2}(t) & L_{3,3}(t) \end{pmatrix}$$

Gambar 2.4. Learning Matrix

Pada saat inisial learning matrix akan seperti pada Gambar 2.5.

$$LM(0) = \begin{pmatrix} 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \\ 0.1111 & 0.1111 & 0.1111 \end{pmatrix}$$

Gambar 2.5 Nilai Awal Learning Matrix

Pada awal inisialisasi Learning Matrix semua nilai diberi nilai yang sama dimana jumlah LM < 1. Pemilihan terhadap rasio didasarkan pada learning matrix ini, oleh karena itu pada awal learning matrix semua memiliki kesempatan. Probabilitas ini akan berubah berdasarkan performa mereka setelah seleksi, oleh karena itu tedapat online learning. Apabila rasio memiliki nilai yang lebih bagus daripada nilai fungsi obyektif yang sebelumnya, maka nilai learning matrix akan

diberikan reward berupa dikali dua seperti yang terilhat pada Gambar 2.6, apabila rasio memiliki nilai yang lebih buruk dari pada nilai fungsi obyektif sebelumnya, maka nilai learning matrix akan diberikan punishment berupa dibagi dua seperti yang terlihat pada Gambar 2.7. Setelah reward atau punishment dijalankan maka Learning Matrix akan mengalami normalisasi agar jumlah LM < 1.

$$LM(1) = \begin{pmatrix} 0.1000 & 0.1000 & 0.1000 \\ 0.2000 & 0.1000 & 0.1000 \\ 0.1000 & 0.1000 & 0.1000 \end{pmatrix}$$

Gambar 2.6 Learning Matrix setelah reward

$$LM(1) = \begin{pmatrix} 0.1176 & 0.1176 & 0.1176 \\ 0.0588 & 0.1176 & 0.1176 \\ 0.1176 & 0.1176 & 0.1176 \end{pmatrix}$$

Gambar 2.7 Learning Matrix setelah punishment

Learning akan dilakukan saat algoritma berjalan pada saat sudah berulang sesuai dengan jumlah yang sudah ditentukan, σ harus sudah ditentukan sebelum mengupdate learning matrix. Pada penelitian Tilahun learning iteration atau σ adalah 5. Nilai ini dapat disesuaikan berdasarkan sumber daya yang ada. Meningkatkan nilainya dapat meningkatkan kualitas solusi akan tetapi membutuhkan lebih banyak perhitungan.

### 2.2.6 *Simulated Annealing* (SA)

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. 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.

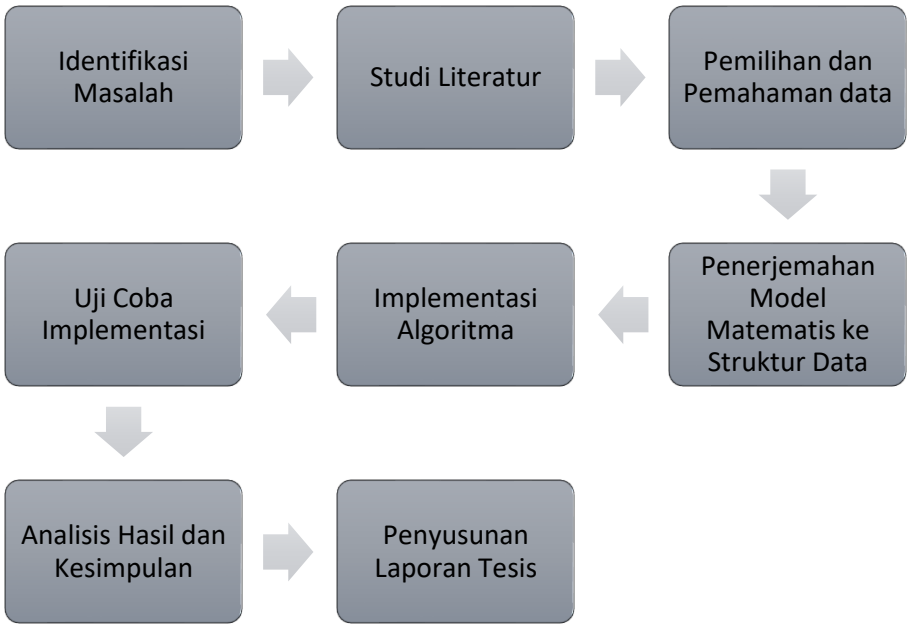
Solusi awal pada *Simulated Annealing* didapatkan dengan acak. Langkah berikutnya yaitu, pencarian pada *search space*. Hasil pencarian tersebut dihitung untuk mendapatkan solusi baru. Apabila solusi baru lebih optimal daripada solusi awal, maka solusi akan diterima sebagai solusi sementara. Jika tidak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

Pseudocode Algoritma Simulated Annealing sebagai berikut :

```
1: Select an initial solution S0
2: Repeat
3:    Randomly select a heuristic h ϵ H
4:    iteration_count = 0
5:    Repeat
6:       iteration_count++
7:       Applying h to S0 , get a new solution S1
8:       δ = current-candidate
9:       if(δ > 0) then S0 = S1
10:      else
11:         Generate a random x uniformly in the range (0,1)
12:         if x>exp(-(δ/t)) then S0 = S1
13:   Until iteration_count = nrep
14:   set t = t * α
15: Until stopping criteria = true;
```

# BAB 3
# METODOLOGI PENELITIAN

Bab ini membahas tahapan penelitian dan penjelasan dari tiap-tiap tahapan penelitian. Secara garis besar, metode yang digunakan dalam penelitian ditunjukkan dengan alur penelitian yang terdapat pada Gambar 3.1. Detail dari metode penelitian ini, dapat dijelaskan sebagai berikut:




Gambar 3. 1 Tahapan Penelitian

## 3.1.Identifikasi Masalah

Tahap identifikasi masalah adalah tahap pertama dalam penelitian ini. Identifikasi masalah digunakan untuk menentukan rumusan masalah, tujuan, kontribusi penelitian. Tahap tersebut telah dibahas pada Bab I. Penelitian ini akan berfokus pada permasalahan penjadwalan ujian pada universitas ETP. Terdapat penelitian menggunakan metode meta-heuristic prey-predator (Tilahun, 2015). Metode tersebut telah dapat diimplementasikan ke berbagai macam persoalan termasuk ETP dengan dataset Carter (Tilahun, 2019) seperti yang sudah dijelaskan

pada Bab II. Pada tahun 2017 terdapat penelitian metode hyper-heuristic prey-predator (Tilahun, 2017) dan mengujinya pada beberapa perhitungan komputasi dan menunjukkan hasil yang bagus. Penelitian tersebut masih bisa dikembangkan agar *best prey* pada *prey predator* tidak terjebak pada local optimum. Penelitian ini akan berfokus pada pengembangan metode hyper-heuristic prey-predator yang telah dikembangkan sebelumnya (Tilahun, 2017) ke dalam ETP dengan dataset ITC 2007.

### 3.2.Studi Literatur

Studi literatur dalam penelitian ini digunakan untuk mengkaji penelitian-penelitian terkait Examination Timetabling dan metode heuristik yang telah digunakan sebelumnya sehingga dapat diketahui perkembangan penelitian pada permasalahan tersebut sebagai bahan untuk analisi gap penelitian dan potensi penelitian selanjutnya. Hal tersebut dilakukan untuk mencari referensi terkait pengembangan metode dan analisis kelebihan dan kekurangan metode yang telah ada. Berdasalkan hasil pengkajian, maka penelitian ini mengusulkan pengembangan metode hyper-heuristic prey-predator dan Simulation Annealing pada Examination dengan menggunakan dataset ITC 2007. Penelitian ini akan diimplementasikan dalam kerangka kerja HyFlex agar dapat dibandingkan performanya. Selain membahas tentang penelitian sebelumnya, penelitian ini juga membahas tentang teori-teori dasar, seperti Examination Timetabling, dataset ITC 2007, Hyper-Heuristic, Simulated Annealing, dan Prey-Predator.

### 3.3.Pemilihan dan Pemahaman Data

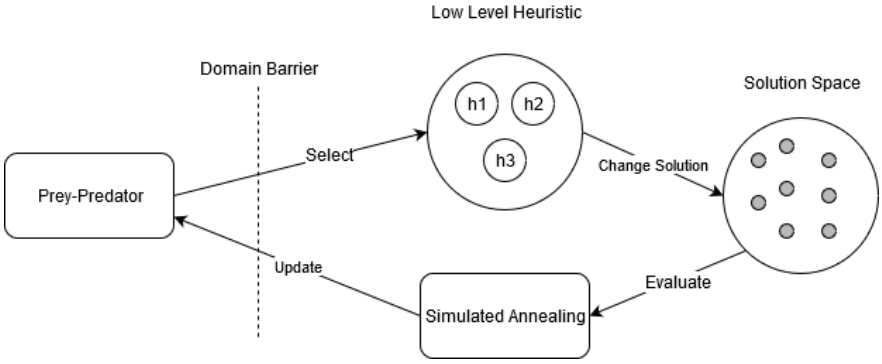
Setelah topik permasalahan dan algoritma telah ditentukan kemudian dilakukan pemilihan data. Data yang dipilih dan digunakan dalam penelitian berupa data *benchmark*, yaitu dataset ITC 2007. Dataset ITC 2007 dipilih karena dataset tersebut lebih mirip dengan persoalan di dunia nyata dibandingkan dengan dataset Carter. Selanjutnya, dilakukan pemahaman dataset ITC yang sudah dibahas pada Bab II pada dasar teori.

### 3.4.Penerjemahan Model Matematis ke Struktur Data

Setelah dilakukan pemahamanan terhadap dataset ITC2007. Kemudian pada tahapan ini, dataset ITC2007 diterjemahkan ke dalam struktur data bahasa pemrograman. Hal ini diperlukan untuk mendukung implementasi algoritma PPA-SA. Hasil yang didapatkan dari tahapan ini adalah struktur data bahasa pemrograman (coding) dari dataset ITC2007.

### 3.5.Implementasi Algoritma

Implementasi algoritma dilakukan dengan menggunakan input dataset ITC 2007. Kemudian program akan membaca file data yang kemudian akan diproses menggunakan algoritma yang digunakan. Didalam algoritma akan membentuk solusi awal secara acak yang kemudian akan dikategorikan menjadi tiga, berdasarkan performa *objective function* yang telah ditentukan. Solusi dengan performa terbaik dinamakan *best prey*, solusi dengan performa terburuk dinamakan *predator* dan sisanya *prey* biasa. Predator bekerja sebagai agen untuk melakukan exploration di solution space dengan cara menakuti prey dan secara acak berlarian. Oleh karena itu, updatenya dilakukan dengan cara berlari mengejar prey yang lemah (prey yang memiliki performa paling rendah). Para prey akan melarikan diri dari predator dan mencoba mengikuti best prey, yang memiliki nilai kelangsungan hidup yang besar. Hal tersebut telah dijelaskan pada Bab II. Setelah solusi terbentuk, maka tugas berikutnya pada algoritma simulated annealing. Simulated Annealing akan diimplementasikan pada *acceptance criteria* dimana algoritma tersebut menentukan keputusan diterima atau tidaknya solusi. Penggunaan algoritma simulated annealing memungkinkan *diversification*, yaitu dapat menerima solusi yang lebih buruk agar tidak terjebak dalam local optimum. Implementasi dilakukan menggunakan framework HyFlex. Pada gambar 3.2 dijelaskan tahap awal implementasi algoritma PPA-SA.




***Gambar 0.2 Framework PPA-SA***

Algoritma *Prey-Predator* akan diimplementasikan menggunakan empat pendekatan *move* satu slot, *swap* satu slot, *move* dua slot, dan *swap* dua slot. Kemudian move acceptance *Simulated Annealing* akan digunakan. *Simulated Annealing* akan menerima solusi yang lebih baik dan juga menerima solusi yang lebih buruk dengan perhitungan proses annealing. Penggunaan algoritma *Prey-Predator* dengan *Simulated Annealing* bertujuan agar best prey pada *Prey-Predator* tidak terjebak pada *local optimum*.

## 3.6.Uji Coba Implementasi

Tahapan uji coba merupakan tahap eksperimen dari implementasi algoritma hyperheuristik. Tahap ini dilakukan untuk memastikan aplikasi dapat berjalan dan menguji algoritma yang digunakan. Uji coba pada permasalahan dilakukan pada framework HyFlex. Terdapat tahapan dalam uji coba ini, yang pertama menentukan lingkungan uji coba, dimana peralatan atau spesifikasi komputer yang digunakan dalam tahap ini dideskripsikan. Kemudian penentuan algoritma yang digunakan untuk uji coba, untuk uji coba ini akan digunakan metode Prey-Predator dan Simulated Annealing, metode Hill Climbing dipilih sebagai pembanding antar metode. Penentuan parameter seperti batasan waktu juga dimasukkan ke dalam uji coba ini.

### 3.7.Analisis Hasil dan Kesimpulan

Tahap ini dilakukan setelah tahap uji coba selesai. Tahap ini digunakan untuk mengukur performa metode Prey-Predator dan Simulated Annealing yang dilakukan. Analisis dilakukan dengan mengamati hasil dari pengujian Prey-Predator dan Simulated Annealing terhadap dataset ITC 2007. Pengamatan dilakukan pada eksperimen dan melakukan perbandingan hasil. Perbandingan hasil dengan metode sebelumnya juga akan dilakukan agar dapat mengukur performa dari metode yang sedang diteliti.

### 3.8.Penyusunan Laporan Tesis

Penyusunan laporan dilakukan dengan cara mengumpulkan semua dokumentasi masukan, proses, dan keluaran dari penelitian. Hasil berupa Buku Laporan Tesis diharapkan dapat menjadi rujukan dan dapat dikembangkan pada penelitian selanjutnya.

*Halaman ini sengaja dikosongkan*

# BAB 4
# ALGORITMA SR-SA, SR-HC, dan PREY PREDATOR

Bab ini akan menjelaskan mengenai penelitian yang telah dilakukan dan akan dikembangkan pada penelitian ini. Adapun pembahasan algoritma *Simple Random – Simulated Annealing*, *Simple Random – Hill Climbing*, dan *Prey-Predator* mengenai eksperimen yang telah dilakukan, hasil, beserta kesimpulannya.

## 4.1.Penjelasan Algoritma

Riset pendahuluan yang telah dilakukan dengan *problem domain* yang sama yaitu *Examination Timetabling Problem* menggunakan dataset ITC 2007. Algoritma yang digunakan adalah algoritma *SR-HC* (Simple Random – Hill Climbing), *SR-SA* (Simple Random – Simulated Annealing), dan *Prey-Predator*. Iterasi *(time limit)* yang digunakan pada penelitian ini sebanyak 300000 milidetik atau 300 detik. Algoritma *SR-HC* digunakan sebagai teknik standar yang digunakan pada teknik optimasi, sedangkan kedua algoritma lainnya dipilih untuk penelitian teknik optimasi.

Pada penelitian yang akan dilakukan, dataset ITC 2007 akan digunakan sebagai dalam penyelesaian masalah *Examination Timetabling Problem.* Karena dataset ini merupakan dataset yang cukup kompleks dibanding dengan *benchmark* dataset ETP yang lain.

## 4.2.Lingkungan Uji Coba

Pada subbab Lingkungan Uji Coba menjelaskan terkait lingkungan pengujian saat melakukan implementasi penelitian terkait optimasi penjadwalan mata kuliah. Spesifikasi perangkat keras yang digunakan dalam implementasi ditunjukkan pada Tabel 4.1.

*Tabel 4. 1 Lingkungan Perangkat Keras Uji Coba*

| Perangkat Keras | Spesifikasi |
|---|---|
| **Jenis** | ASUS ROG STRIX GL503GE |
| **Processor** | Intel Core i7-8750H 2.2GHz |
| **RAM** | 24 GB |
| **Hard Disk Drive** | 1 TB |

Tabel 4.2 menunjukkan spesifikasi perangkat lunak yang digunakan dalam pengerjaan penelitian, termasuk sistem operasi beserta framework yang digunakan.

*Tabel 4. 2 Lingkungan Perangkat Lunak Uji Coba*

| Perangkat Lunak | Fungsi |
|---|---|
| **Windows 7 64 bit** | Sistem Operasi |
| **Netbeans 8.0** | Implementasi Algoritma |
| **HyFlex** | Framework *Hyper-heuristic* |

### 4.3.Hasil Eksperimen Dataset ITC 2007

Pada sub bab ini membahas mengenai hasil eksperimen dataset ITC 2007 dengan menggunakan algoritma standart, yaitu algoritma SR-HC, SR-SA dan PPA. Eksperimen dengan dataset ITC 2007 sebelumnya tidak pernah dilakukan menggunakan aplikasi yang telah dikembangkan di riset pendahuluan. Eksperimen yang akan dilakukan menggunakan parameter *default* yaitu menggunakan *time limit* 300000 milidetik atau 300 detik. Dengan adanya eksperimen dataset ITC 2007 diharapkan dapat membuat aplikasi mencapai hasil yang lebih optimum dan aplikasi menjadi lebih *general* dalam *Examination Timetabling*. Kompleksitas mengenai dataset ITC 2007 dapat dilihat pada subbab 2.2.3. ETP Pada Dataset ITC 2007. Tabel 4.3 memperlihatkan nilai yang dibentuk menggunakan solusi awal. Tabel 4.3 dan Tabel 4.4, tabel 4.5, dan Tabel 4.6 menunjukkan hasil eksperimen dataset ITC dengan aplikasi riset yang sebelumnya dikembangkan.

*Tabel 4. 3 Hasil Skenario Uji Coba dengan pembentukan solusi awal*

| PERCOBAAN | INSTANCE 1 | INSTANCE 2 | INSTANCE 3 | INSTANCE 4 | INSTANCE 5 | INSTANCE 6 | INSTANCE 7 | INSTANCE 8 |
|---|---|---|---|---|---|---|---|---|
| | EXAM 1 | EXAM 2 | EXAM 3 | EXAM 4 | EXAM 5 | EXAM 6 | EXAM 7 | EXAM 8 |
| 1 | 32353 | 37039 | 103838 | NA | 136049 | 56715 | 70515 | 156305 |
| 2 | 32432 | 39872 | 93238 | NA | 144042 | 51650 | 68644 | 132968 |
| 3 | 28795 | 37486 | 119529 | NA | 134795 | 57665 | 66394 | 148569 |
| 4 | 32802 | 43717 | 100841 | NA | 150648 | 51570 | 70753 | 115102 |
| 5 | 30417 | 33459 | 96380 | NA | 130856 | 58885 | 65895 | 116972 |
| 6 | 30532 | 36931 | 109290 | NA | 124175 | 57117 | 66257 | 160778 |
| 7 | 29293 | 37330 | 117696 | NA | 137924 | 56065 | 68684 | 107610 |
| 8 | 32102 | 40180 | 109272 | NA | 136202 | 47360 | 69861 | 116716 |
| 9 | 30175 | 53750 | 102596 | NA | 147253 | 51040 | 55875 | 178316 |
| 10 | 32215 | 39071 | 91278 | NA | 157996 | 54610 | 62342 | 94781 |
| | | | | | | | | |
| **AVG** | **31111** | **39883** | **104385** | **NA** | **139994** | **54267** | **66522** | **132811** |
| *BEST* | 28795 | *104385* | *91278* | NA | *124175* | *47360* | *55875* | *94781* |

Keterangan :

(NA) Not Available, Gagal menghasilkan solusi awal hingga batas waktu yang telah ditentukan

(AVG) Average, Nilai rata-rata hasil 10 eksperimen

(Best) Nilai terbaik dari hasil 10 eksperimen

*Tabel 4. 4* Hasil Skenario Uji Coba dengan Algoritma SR-HC-HH

| PERCOBAAN | INSTANCE 1 | INSTANCE 2 | INSTANCE 3 | INSTANCE 4 | INSTANCE 5 | INSTANCE 6 | INSTANCE 7 | INSTANCE 8 |
|---|---|---|---|---|---|---|---|---|
| | EXAM 1 | EXAM 2 | EXAM 3 | EXAM 4 | EXAM 5 | EXAM 6 | EXAM 7 | EXAM 8 |
| 1 | 29023 | 39930 | 98521 | NA | 147047 | 53375 | 62038 | 117933 |
| 2 | 31037 | 36419 | 93852 | NA | 96288 | 51655 | 70147 | 159235 |
| 3 | 28744 | 36482 | 97810 | NA | 153202 | 54635 | 61542 | 115821 |
| 4 | 29324 | 38873 | 107308 | NA | 124207 | 63325 | 74064 | 145700 |
| 5 | 35758 | 45654 | 102527 | NA | 134480 | 59465 | 64212 | 137839 |
| 6 | 31394 | 35819 | 121358 | NA | 121359 | 57540 | 66823 | 127383 |
| 7 | 32216 | 41221 | 105952 | NA | 142838 | 54925 | 60220 | 107002 |
| 8 | 29424 | 37861 | 93012 | NA | 123624 | 49620 | 76214 | 119590 |
| 9 | 31029 | 42922 | 101111 | NA | 135047 | 52235 | 70796 | 132183 |
| 10 | 33595 | 38641 | 106768 | NA | 120646 | 54285 | 72852 | 131197 |
| | | | | | | | | |
| **AVG** | **31154** | **39322** | **102826** | **NA** | **129837** | **55106** | **67890** | **129388** |
| *BEST* | 28744 | *35819* | *93012* | NA | *96288* | *49620* | *60220* | *107002* |

Keterangan :

(NA) Not Available, Gagal menghasilkan solusi awal hingga batas waktu yang telah ditentukan

(AVG) Average, Nilai rata-rata hasil 10 eksperimen

(Best) Nilai terbaik dari hasil 10 eksperimen

*Tabel 4. 5 Hasil Skenario Uji dengan Algoritma SR-SA-HH*

| PERCOBAAN | INSTANCE 1 | INSTANCE 2 | INSTANCE 3 | INSTANCE 4 | INSTANCE 5 | INSTANCE 6 | INSTANCE 7 | INSTANCE 8 |
|---|---|---|---|---|---|---|---|---|
| | EXAM 1 | EXAM 2 | EXAM 3 | EXAM 4 | EXAM 5 | EXAM 6 | EXAM 7 | EXAM 8 |
| 1 | 32042 | 45973 | 102473 | NA | 100953 | 46610 | 64148 | 106697 |
| 2 | 29970 | 38655 | 92886 | NA | 113029 | 49670 | 55374 | 91086 |
| 3 | 29178 | 41546 | 86374 | NA | 107209 | 56980 | 67243 | 126148 |
| 4 | 28751 | 42108 | 101171 | NA | 117832 | 55355 | 58651 | 109976 |
| 5 | 32859 | 39888 | 100291 | NA | 110692 | 46820 | 59087 | 109900 |
| 6 | 31609 | 30298 | 100693 | NA | 125060 | 50490 | 61624 | 134497 |
| 7 | 33319 | 33622 | 97801 | NA | 111191 | 55805 | 68456 | 111332 |
| 8 | 28711 | 41579 | 97685 | NA | 115359 | 57175 | 58555 | 122759 |
| 9 | 28596 | 38099 | 99542 | NA | 115898 | 54390 | 65478 | 142875 |
| 10 | 32358 | 37711 | 110148 | NA | 118803 | 45290 | 70354 | 136452 |
| | | | | | | | | |
| **AVG** | **30739** | **38947** | **98906** | **NA** | **113602** | **45290** | **62897** | **119172** |
| *BEST* | 28596 | *30298* | *86374* | NA | *100953* | *51858* | *55374* | *91086* |

Keterangan :

(NA) Not Available, Gagal menghasilkan solusi awal hingga batas waktu yang telah ditentukan

(AVG) Average, Nilai rata-rata hasil 10 eksperimen

(Best) Nilai terbaik dari hasil 10 eksperimen

*Tabel 4. 6 Hasil Skenario Uji Coba dengan Algoritma PPA-HH*

| PERCOBAAN | INSTANCE 1 | INSTANCE 2 | INSTANCE 3 | INSTANCE 4 | INSTANCE 5 | INSTANCE 6 | INSTANCE 7 | INSTANCE 8 |
|---|---|---|---|---|---|---|---|---|
| | EXAM 1 | EXAM 2 | EXAM 3 | EXAM 4 | EXAM 5 | EXAM 6 | EXAM 7 | EXAM 8 |
| 1 | 30191 | 46261 | 97221 | NA | 93872 | 57040 | 69452 | 106215 |
| 2 | 30057 | 42811 | 103102 | NA | 82999 | 51560 | 61281 | 88680 |
| 3 | 30443 | 38169 | 98589 | NA | 94516 | 51150 | 69289 | 92203 |
| 4 | 29652 | 39328 | 108125 | NA | 91047 | 49795 | 61841 | 111024 |
| 5 | 30811 | 35086 | 110887 | NA | 91146 | 52440 | 63710 | 79728 |
| 6 | 33857 | 37005 | 92229 | NA | 79923 | 50365 | 68123 | 97175 |
| 7 | 30844 | 42454 | 101005 | NA | 99963 | 54710 | 68545 | 91881 |
| 8 | 32509 | 42243 | 92948 | NA | 97007 | 57180 | 58095 | 86336 |
| 9 | 29317 | 39280 | 96937 | NA | 79718 | 60875 | 61622 | 123868 |
| 10 | 31935 | 39902 | 101875 | NA | 114399 | 55625 | 71131 | 113876 |
| | | | | | | | | |
| **AVG** | **30961** | **40253** | **100291** | NA | **92549** | **54069** | **65308** | **99098** |
| *BEST* | 29317 | *35086* | *92229* | NA | *79718* | *49795* | *58095* | *79728* |

Keterangan :

(NA) Not Available, Gagal menghasilkan solusi awal hingga batas waktu yang telah ditentukan

(AVG) Average, Nilai rata-rata hasil 10 eksperimen

(Best) Nilai terbaik dari hasil 10 eksperimen

## 4.4.Perbandingan Hasil Eksperimen Dataset ITC 2007

Perbandingan hasil eksperimen penelitian dataset ITC 2007 dilakukan terhadap hasil yang telah didapatkan. Dapat dilihat pada tabel 4.3, terdapat 7 instances dari 8 instances yang solusinya berhasil tampil, yaitu *exam1*, *exam2, exam3, exam5, exam6, exam7* dan *exam8.* Sisanya yaitu *exam4* telah kehabisan waktu hanya untuk melakukan inisialisasi awal solusi oleh karena itu exam4 tidak memiliki nilai. Pada tabel 4.4 percobaan melakukakaan Hill Climbing sebagai move acceptance pada examination timetabling berhasil dengan baik, akan tetapi algoritma Simple Random - Simulated Annealing yang ditunjukkan pada Tabel 4.5 menunjukkan hasil yang lebih baik daripada Simple Random – Hill Climbing pada semua instance, kecuali satu instance dikarenakan melebihi batas waktu.

*Tabel 4. 7 Hasil Skenario Uji Coba perbandingan algoritma*

| EXAM | Solusi Awal | SR-HC-HH | SR-SA-HH | PPA-HH | MUKLASON 2017 | | | (Müller, 2009) | (Demeester et al., 2012) | (Rahman et al., 2014) | (Alzaqebah and Abdullah, 2014) | (Burke et al., 2014) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | SR-GD-HH | RL-GD-HH | SA-GD-HH | | | | | |
| **1** | 31111 | 31154 | 30793 | 30961 | 6579 | 7019 | 5809 | *4370* | 6060 | 5231 | 5328 | 6235 |
| **2** | 39883 | 39322 | 38947 | 40253 | 584 | 535 | 490 | *400* | 515 | 433 | 512 | 2974 |
| **3** | 104385 | 102826 | 98906 | 100291 | 11153 | 11592 | 10819 | *10049* | 23580 | 9265 | 10178 | 15832 |
| **4** | NA | NA | NA | NA | 13233 | 21992 | 14100 | 18141 | NA | 17787 | 16465 | 35106 |
| **5** | 139994 | 129837 | 113602 | 92459 | 3658 | 4610 | 3596 | *2988* | 4855 | 3083 | 3624 | 4873 |
| **6** | 54267 | 55106 | 51858 | 54069 | 26515 | 28130 | *26075* | 26950 | 27605 | 26060 | 26240 | 31756 |
| **7** | 66522 | 67890 | 62897 | 65308 | 5145 | 5151 | 5185 | *4213* | 6065 | 10712 | 4562 | 11562 |
| **8** | 132811 | 129388 | 119172 | 99098 | 9348 | 11405 | 9180 | *7861* | 9038 | 12713 | 8043 | 20994 |

Keterangan :

(NA) Not Available, Gagal menghasilkan solusi awal hingga batas waktu yang telah ditentukan

(AVG) Average, Nilai rata-rata hasil 10 eksperimen

(Best) Nilai terbaik dari hasil 10 eksperimen

## 4.5.Kesimpulan

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

1. Algoritma sederhana yang digunakan pada aplikasi yang telah dikembangkan yaitu *Simple Random – Hill Climbing* berbasis *Hyper-heuristics* mampu menyelesaikan permasalahan penjadwalan pada *domain Examination Timetabling* Problem pada dataset ITC 2007, namun masih belum optimal. Pada algoritma *Simple Random – Simulated Annealing* memiliki nilai yang lebih baik daripada *Simple Random – Hill Climbing*, hal ini disebabkan karena *Hill Climbing* dapat terjebak pada local optimum, sedangkan algoritma *Simulated Annealing* mampu mengatasi hal tersebut dengan proses annealing. Algoritma *Prey-Predator* sendiri memiliki nilai terbaik pada 2 exam, yaitu exam 5 dan exam 8 dengan perbedaan nilai yang cukup jauh apabila dibandingkan dengan *Simple Random – Hill Climbing* dan *Simple Random – Simulated Annealing*.
2. Pengubahan parameter pada skenario uji coba juga dapat mempengaruhi hasil solusi yang lebih optimum. Parameter yang dapat diubah dalam penelitian ini adalah jumlah iterasinya (*time limit*) dan algoritma untuk seleksi *Low Level Heuristik.*

## 4.6.Penelitian yang Akan Dilakukan Kedepannya

Dapat dikatakan bahwa permasalahan yang dihadapi saat melakukan eksperimen dengan dataset ITC 2007 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. Parameter yang akan ditunning adalah strategi seleksi LLH dan move acceptance dalam hyper-heuristic. Penelitian selanjutnya akan menggunakan algoritma *Prey-Predator* sebagai seleksi LLH, kemudian untuk move acceptance akan menggunakan algoritma *Simulated Annealing*. Kedua algoritma tersebut yang akan digunakan saat fase optimasi pada metode hyper-heuristic.

*Halaman ini sengaja dikosongkan*

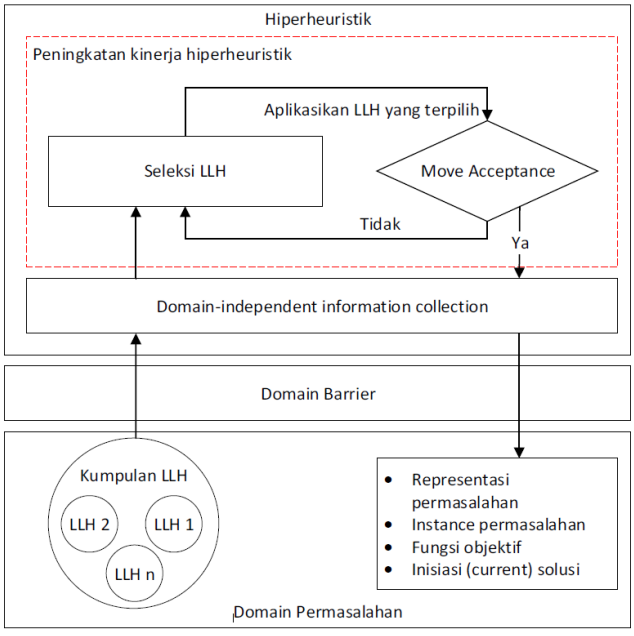
# BAB 5
# DESAIN DAN IMPLEMENTASI

Dalam tahap desain ini dilakukan perancangan algoritma untuk menentukan strategi pemilihan LLH dan implementasi *move acceptance* yang tepat dalam menyelesaikan permasalahan ETP menggunakan metode hiperheuristik.

## 5.1.Desain Hiperheuristik

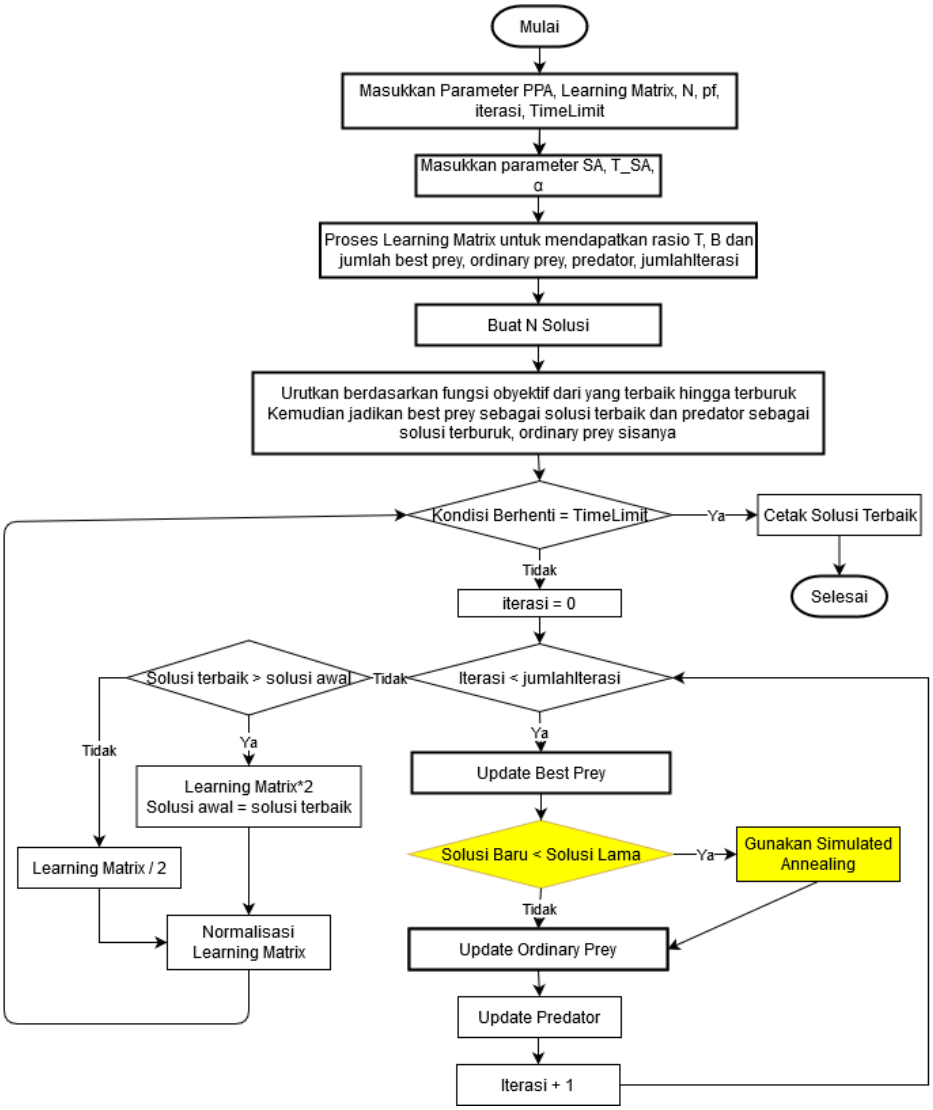
Pada umumnya dalam metode hiperheuristik terdapat dua tingkatan yaitu level atas (high) dan level bawah. Hiperheuristik pada level atas atau biasa disebut High Level Heuristic (HLH) secara umum sebuah proses yang akan berlangsung selama iterasi tertentu atau selama kondisi terpenuhi akan memilih LLH yang sudah disiapkan. LLH yang terpilih akan membangun solusi baru, kemudian solusi tersebut akan diputuskan diterima atau tidaknya oleh *move acceptance*. Jika diterima, solusi tersebut akan menjadi acuan pada iterasi berikutnya untuk dirubah oleh LLH, tetapi jika tidak diterima maka akan berlangsung ke iterasi berikutnya untuk mencari LLH.

Strategi HLH terdiri dari dua mekanisme, yaitu seleksi LLH dan move *acceptance*. Secara umum gambar 5.1 dijelaskan kesuluruhan strategi HLH.




Gambar 5. 1 Kerangka Hiperheuristik

Dalam menentukan strategi baru untuk HLH dilakukan dengan mengkombinasikan dua metode, yaitu Prey-Predator sebagai strategi pemilihan LLH dan Simulated Annealing sebagai *move acceptance*. Pada desain PPA-SA ini dimulai dengan inisialisasi solusi untuk mendapatkan solusi awal. Solusi awal ini kemudian akan diproses hingga mendapatkan nilai fungsi obyektif. Desain algoritma PPA-SA dapat dilihat pada gambar 5.2.




Gambar 5. 2 Desain Prey Predator – Simulated Annealing

### 5.1.1. Pengaturan Parameter

Pada penelitian ini, setiap instance akan di berikan batas waktu sekitar 300 detik dan dieksekusi sebanyak 10 kali. Dalam framework Hyflex menggunakan parameter milidetik, oleh karena itu dimasukkan nilai 300000 atau lima menit.

Sedangkan pada hiperheuristik yang diusulkan membutuhkan beberapa parameter, seperti prosentase perbandingan yang akan digunakan pada Learning Matriks, probability of follow up (Pf), jumlah solusi, dan iterasi. Dalam penelitian ini nilai prosentase yang digunakan adalah 0.25, 0.5, dan 0.75 (Tilahun, 2017). Jumlah solusi (N) menggunakan 20 solusi (Tilahun, 2019), yang digunakan untuk *best prey*, *predator* dan *ordinary prey* sebagai agen *exploration* dan *exploitation*. Parameter selanjutnya adalah iterasi PPA akan berjalan dengan nilai 50 (Tilahun, 2019). Probability of follow up (Pf), digunakan sebagai probabilitas *ordinary prey* akan mengikuti *best prey* dimana nilainya adalah 0.2 dan 0.6 (Tilahun, 2019). Daftar parameter yang digunakan bisa dilihat pada tabel 5.1.

*Tabel 5. 1 Daftar parameter yang digunakan*

| No. | Paramater | Nilai |
|---|---|---|
| 1 | Jumlah Solusi (N) | 20 |
| 2 | Rasio best prey dan predator terhadap N (T) | 0.25, 0.5, 0.75 |
| 3 | Rasio best prey terhadap T (B) | 0.25, 0.5, 0.75 |
| 4 | Iterasi PPA | 50 kali |
| 5 | Probability of Follow-up (pf) | 0.2 dan 0.6 |

Sedangkan rangkaian LLH yang digunakan pada permasalahan penjadwalan ujian ini menggunakan enam LLH. Rincian LLH dapat dilihat pada tabel 5.2

Tabel 5. 2 Daftar LLH yang digunakan

| No. | Nama LLH | Deskripsi |
|---|---|---|
| 1 | Swap Exam | ujian yang dipilih secara acak ditukarkan ke waktu dan ruang yang baru secara acak |
| 2 | Swap Period | ujian yang dipilih secara acak ditukarkan ke waktu baru secara acak |
| 3 | Swap Room | ujian yang dipilih secara acak ditukarkan ke kamar baru secara acak |
| 4 | Change Period | ujian yang dipilih secara acak akan diganti ke waktu yang baru secara acak |
| 5 | Change Room | ujian yang dipilih secara acak akan diganti ke ruang yang baru secara acak |
| 6 | Change Period Room | ujian yang dipilih secara acak akan diganti ruang dan waktunya secara acak |

## 5.2.Implementasi HyFlex

### 5.2.1. Kelas Eksekusi (Method Main)

Kelas ini untuk menunjukkan cara menjalankan hyper-heuristic menggunakan domain masalah yang telah dipilih. Framework hyflex terdiri dari dua abstrak, yaitu ProblemDomain dan HyperHeuristic. Kelas ProblemDomain menyediakan berbagai elemen yang dapat dengan mudah diakses dan dikelola menggunakan satu method atau lebih. Sedangkan HyperHeuristic class dirancang untuk memungkinkan algoritma yang mengimplementasikan kelas ini, untuk dibandingkan dan menjadi tolak ukur di satu atau lebih problem domain yang tersedia (misalnya digunakan dalam kompetisi).

```
3 import AbstractClasses.HyperHeuristic;
4 import AbstractClasses.ProblemDomain;
5 import itc2007ExamProblems.ITC2007Exam;
```

Kode Program 5.1 Hyflex Class

Pada baris ke lima pada Kode Program 5.1 merupakan bagian awal saat pengkodean yaitu deklarasi library yang dibutuhkan saat uji coba. Baris kedua dan ketiga merupakan pemanggilan fungsi hiperheuristik yang akan digunakan saat uji coba yang berada pada folder atau package AbstractClasses. Baris kelima

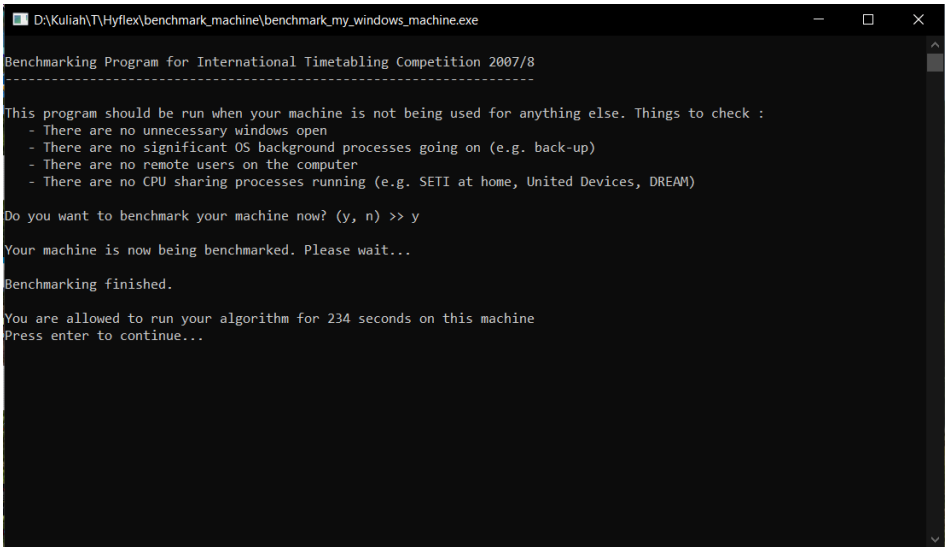
merupakan pemanggilan domain yang digunakan pada uji coba, dimana disini adalah kelas ITC 2007.

```
20     public static void main(String[] args) {
21
22         ProblemDomain problem = new ITC2007Exam(6871);
23         HyperHeuristic hyper_heuristic_object = new PPSA3(5678);
24         problem.loadInstance(1);
25         hyper_heuristic_object.setTimeLimit(300000);
26         hyper_heuristic_object.loadProblemDomain(problem);
27         hyper_heuristic_object.run();
28         System.out.println("best solution:" +
hyper_heuristic_object.getBestSolutionValue());
29         System.out.println("best solution:\n" +
problem.bestSolutionToString());
30
31     }
```

Kode 5.2 Main Method

Pada kode 5.2 menampilkan bagaimana program digunakan. Pada baris ke-22 dan 23 kelas abstrak ProblemDomain dan HyperHeuristic dideklarasikan, tujuannya adalah agar kelas ITC2007Exam mendapatkan fungsi turunan dari kelas abtrak ProblemDomain, begitu juga dengan kelas PPASA2 dapat menggunakan fungsi yang berada pada kelas abstrak Hyperheuristic. Nomor yang digunakan pada parameter kelas ITC2007Exam dan kelas PPASA2 bertujuan sebagai *seed random* atau digunakan sebagai *pseudorandom*, jadi setiap kali running nomor yang teracak selalu berada pada urutan yang sama. Pada baris 22 kita bisa melihat bahwa permasalahan yang dihadapi adalah ITC2007Exam, sedangkan pada baris 23 kita bisa melihat strategi yang digunakan berada pada kelas PPSA3.

Pada baris ke-24 bertujuan untuk memuat data permasalahan yang artinya program sedang menjalankan permasalahan pada instance indeks satu. Pada baris ke-25 merupakan fungsi yang digunakan untuk mengatur waktu dengan parameter milliseconds (ms). Untuk dataset ITC 2007 telah disediakan program untuk menghitung waktu yang harus dijalankan saat menjalankan program. Untuk aplikasi bisa didownload pada http://www.cs.qub.ac.uk/itc2007/. Setelah aplikasi dijalankan, maka tampilan yang muncul seperti pada Gambar 5.3.




Gambar 5. 3 Hasil benchmark time limit

Pada baris ke-26 merupakan penetapan objek dari abstrak kelas ProblemDomain diatur ke objek dari abstrak kelas HyperHeuristic atau dengan kata lain pengarahan terhadap objek hyperheuristic untuk mengenali atau menjadikan objek problem domain sebagai referensi. Bagian ini merupakan langkah yang dijalankan setelah data *instance problem* berhasil dimuat dan batas waktu telah ditetapkan. Pada baris ke-27 terdapat fungsi run(), fungsi tersebut dijalankan untuk mengeksekusi proses. Pada baris ke-28 menampilkan fungsi obyektif solusi terbaik.

### 5.2.2. Data Instance Problem

Dataset ITC 2007 memiliki informasi mengenai exam, peserta ujian, period, room dan penalti, period dan room hard constraint, dan informasi soft constraint serta bobot yang biasa disebut dengan Institutional Weigthings. Data tersebut berupa teks seperti pada Gambar 5.4. Cara membacanya pada kolom pertama berisi informasi tentang durasi yang dimiliki pada setiap ujian, kemudian angka setelah itu menampilkan id peserta ujian. Kelas yang berfungsi untuk membaca dataset tersebut berada pada kelas ITC2007ExamInstance.

```
[Exams:607]
195, 2829, 2864, 2882, 4466, 4571, 4878, 4913,
135, 2974, 4950, 5061, 5138, 5139, 5140, 5141,
120, 270, 614, 668, 692, 716, 1546, 1865, 2431,
135, 33, 113, 285, 288, 290, 692, 715, 1546, 18
135, 31, 261, 285, 288, 290, 303, 304, 328, 614
135, 33, 112, 127, 140, 144, 251, 472, 626, 671
135, 251, 295, 297, 298, 667, 671, 673, 701, 71
```

Gambar 5. 4 Contoh data ITC 2007

### 5.2.3. Fase Konstruksi

Awalnya HyFlex digunakan untuk mendukung kompetisi penelitian internasional dalam tantangan pencarian heuristik lintas domain pada tahun 2011 atau dikenal dengan CheSC (Cross-Domain Heuristic Search Challenge) 2011. Mengingat versi Hyflex tersebut belum terdapat implementasi Examination Timetabling Problem (ETP) sebagai salah satu problem domain, maka domain tersebut ditambahkan sebagai problem domain yang baru pada Hyflex.

Sebagaimana telah disebutkan sebelumnya bahwa, terdapat beberapa hal yang dibutuhkan ketika mengimplementasikan ETP sebagai problem domain pada Hyflex. Salah satu hal yang perlu ditambahkan adalah metode heuristic pembentukan inisial solusi.

Pada dataset ini, pembentukan inisial solusi diadopsi dari Carter dkk (1996), largest degree graph colouring, diimplementasikan pada dataset ini. Dimana exam dengan constraint terbanyak akan ditempatkan terlebih dahulu, dan penempatan terhadap exam dengan constraint sedikit akan dilakukan di akhir. Jika solusi yang layak tidak ditemukan maka exam yang mengalami konflik, keduanya, akan dipindahkan (secara random) dan dihapus bersama dengan exam lain yang telah ditempatkan sebelumnya. Setalah itu, daftar exam yang telah dihapus tersebut, akan ditempatkan ulang sesuai dengan prosedur awal. Proses akan berhenti ketika seluruh exam telah berhasil ditempatkan.

Selain itu, komponen lain yang perlu ditambahkan kedalam Hyflex adalah pendefinisian daftar low level heuristic (LLH). Pada penelitian ini, daftar LLH berikut digunakan untuk mendukung atau meningkatkan performa dari algoritma. Daftar LLH tersebut diadaptasi dari (Muklason, 2017).

Sedangkan pembentukan inisial solusi pada dataset ITC 2007 dimulai dengan menempatkan exam yang memiliki jumlah peserta ujian terbanyak terlebih dahulu. Exam ditempatkan pada sebuah room dimana faktor daya tampung dan durasi constraint dinilai layak dengan langkah sebagai berikut: langkah pertama akan dilakukan pengecekan apakah sebuah exam dapat ditempatkan pada satu set ruang dan waktu yang telah memiliki isi.

Jika kondisi terpenuhi dan durasi exam tidak lebih panjang daripada durasi slot period, maka exam akan ditempatkan pada set yang pertama kali ditemui. Tetapi jika tidak ditemukan satu set dengan kapasitas daya tampung untuk mengakomodasi suatu exam, maka exam tersebut akan ditempatkan secara random pada room yang kosong. Dengan syarat bahwa durasi dari exam harus setara atau lebih kecil daripada durasi dari slot period room tersebut

### 5.2.4. Fase Improvement Algoritma dengan Strategi Hyperheuristic

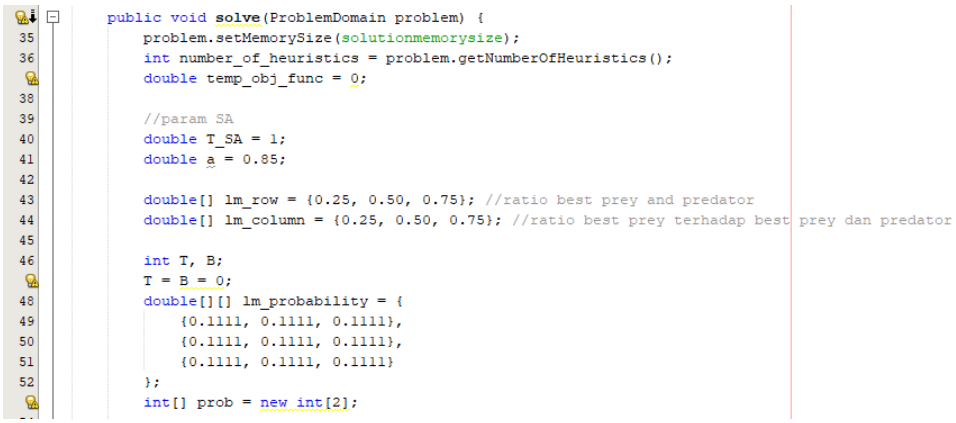
.Pada tahap ini akan dijelaskan mengenai pengkodean yang dilakukan untuk tiap barisnya.

```
20      int solutionmemorysize = 17; //1 main, 10 solusi PPA, 6 direction for best prey
21      int N = 10; //jumlah solusi
22      int rfs = 100; // range from start
23      double[] current_obj_function_values = new double[solutionmemorysize]; //obj_func init
24      double[] sort_obj_function = new double[N]; // obj_func setelah disortir
25      int[] obj_function_num = new int[N]; // obj_func urutan
26      double[] llh_obj_function_values = new double[6];
27      int fixedIteration = 100;
28      double pf = 0.5; //probability of follow up
29
30      public PPASA2(long seed) {
31          super(seed);
32      }
```

Gambar 5. 5 Kode inisialisasi data

Pada Gambar 5.5. variable solutionmorysize digunakan mengisi jumlah solusi yang nantinya akan digunakan diantaranya 1 solusi utama, 10 solusi Prey-Predator, dan 6 solusi akan digunakan untuk memperbarui solusi best prey pada Prey-Predator. Variabel N digunakan sebagai jumlah solusi yang nantinya akan dibagi menjadi best prey, ordinary prey, dan predator. Variabel current_obj_function_values digunakan untuk menyimpan fungsi obyektif pada masing-masing solusi. Sedangkan variable sort_obh_function digunakan untuk menyimpan solusi yang sudah diurutkan berdasarkan yang paling baik ke buruk. Variabel obj_function_num sendiri digunakan untuk menyimpan urutan

berdasarkan id pada variable current_obj_function_values yang sudah diurutkan. Variabel llh_obj_function_values digunakan untuk menyimpan solusi yang digunakan untuk mengupdate best prey. Variable fixedIteration digunakan sebagai jumlah perulangan yang digunakan pada aplikasi. Sedangkan variable pf merupakan singkatan dari *probability of follow*, dimana variabel tersebut digunakan sebagai kemungkinan *ordinary prey* akan mengikuti *best prey* dan kemungkinan predator akan mengikuti *ordinary prey*.



```
35      public void solve(ProblemDomain problem) {
36          problem.setMemorySize(solutionmemorysize);
37          int number_of_heuristics = problem.getNumberOfHeuristics();
38          double temp_obj_func = 0;
39
40          //param SA
41          double T_SA = 1;
42          double a = 0.85;
43
44          double[] lm_row = {0.25, 0.50, 0.75}; //ratio best prey and predator
45          double[] lm_column = {0.25, 0.50, 0.75}; //ratio best prey terhadap best prey dan predator
46
47          int T, B;
48          T = B = 0;
49          double[][] lm_probability = {
50              {0.1111, 0.1111, 0.1111},
51              {0.1111, 0.1111, 0.1111},
52              {0.1111, 0.1111, 0.1111}
53          };
54          int[] prob = new int[2];
```

Gambar 5. 6 Kode inisialisasi PPA-SA

Pada baris ke-35 merupakan fungsi untuk memperbesar ukuran dari tempat penyimpanan solusi. Pada baris ke-36 merupakan deklarasi jumlah heuristik yang digunakan. Pada baris ke-40 dan 41 merupakan deklarasi parameter yang digunakan untuk Simulated Annealing. Pada baris ke-43 dan 44 merupakan deklarasi rasio yang digunakan pada Prey-Predator. Variabel lm_probability merupakan variabel yang digunakan untuk menyimpan Learning Matrix dan hasil pengacakannya akan disimpan pada variabel prob.

```
56          prob = get_LM_prob(lm_probability);
57          T = (int) (lm_row[prob[0]] * N);
58          B = (int) (lm_column[prob[1]] * T);
59          System.out.println("T : " + T);
60          System.out.println("B : " + B);
61
62          int best_prey = B;
63          int predator = T - B;
64          int ordinary_prey = N - (T);
65
66          System.out.println("best prey : " + best_prey);
67          System.out.println("predator : " + predator);
68          System.out.println("ordinary prey : " + ordinary_prey);
69
70          Random r = new Random();
71          problem.initialiseSolution(0);
72          current_obj_function_values[0] = problem.getFunctionValue(0);
73          sort_obj_function[0] = problem.getFunctionValue(0);
74          for (int sl = 1; sl <= N; sl++) {
75              problem.copySolutionRl(0, sl);
76              current_obj_function_values[sl] = problem.getFunctionValue(sl);
77          }
```

Gambar 5. 7 Kode rasio PPA

Pada baris ke-56 merupakan pemanggilan fungsi yang digunakan memilih secara acak baris dan kolom berdasarkan persentase dari nilai variabel lm_probability. Setelah disimpan ke dalam variabel prob, maka hasil akan dikeluarkan ke variabel T dan B. Contoh, apabila yang terpilih adalah baris 1 dan kolom 2, maka lm_row[prob[0]] bernilai 0.25, maka T bernilai menjadi 0.25 * 10 menjadi 2.5, kemudian dibulatkan menjadi tipe data Integer 2. Sedangkan lm_column[prob[1]] bernilai 0.5, maka B sama dengan 0.5 * 2 menjadi 1. Sedangkan ordinary_prey bernilai N-T menjadi 10-2, maka nilai ordinary prey adalah 8. Baris ke 71, merupakan fungsi untuk membuat inisialisasi solusi ke indeks 0. Pada baris ke-72 fungsi obyektif akan dihitung dan dimasukkan ke dalam current_obj_function_values indeks 0. Pada baris 74 hingga 77 solusi indeks 0 akan di duplikat ke solusi indeks 1 sampai 10. Setelah di duplikat, maka menggunakan *Simple Random* tiap solusi akan di lakukan perulangan hingga 100 kali untuk menciptakan variasi pada solusi.

```
97          //////sort
98          Arrays.sort(sort_obj_function);
            double[] current_obj_function_values_temp = new double[solutionmemorysize];
100         current_obj_function_values_temp = current_obj_function_values;
101         System.out.println("SORTED obj function");
102         for (int x = 1; x <= N; x++) {
103             System.out.println("sort obj func " + x + " : " + sort_obj_function[x - 1]);
104             for (int y = 1; y <= N; y++) {
105                 if (sort_obj_function[x - 1] == current_obj_function_values_temp[y]) {
106                     obj_function_num[x - 1] = y;
107                     current_obj_function_values_temp[y] = Double.MAX_VALUE;
108                     break;
109                 }
110             }
111         }
112
113         System.out.println("sorted number :");
114         for (int i = 0; i < obj_function_num.length; i++) {
115             System.out.println("num " + i + " : " + obj_function_num[i]);
116         }
```

Gambar 5. 8 Kode sorting solusi

Setelah solusi berhasil dibuat langkah selanjutnya adalah pengurutan solusi. Solusi diurutkan berdasarkan nilai fungsi obyektifnya dari yang terbaik dan terburuk. Solusi yang terbaik didapatkan oleh *best prey*, solusi yang paling buruk didapatkan *predator*, sisanya didapatkan oleh *ordinary prey*.

```
//////////////movement best prey
for (int count_best_prey = 0; count_best_prey < best_prey; count_best_prey++) { //until last best prey
    // all six path LLH
    boolean path_available = false;
    double best_obj_function = 0;
    for (int llh = 0; llh < number_of_heuristics; llh++) { //all LLH
        llh_obj_function_values[llh] = problem.applyHeuristic(llh, obj_function_num[count_best_prey], 11 + llh);
        if (llh_obj_function_values[llh] < current_obj_function_values[obj_function_num[count_best_prey]]) {// jika movement lebih bagus
            path_available = true;
        }
    }

    // find best path
    if (path_available == true) {
        int path = find_bestPath(llh_obj_function_values, 0, llh_obj_function_values.length);
        problem.copySolutionRl(11 + path, obj_function_num[count_best_prey]); //copy best path ke best prey yg bersangkutan
        LastStepMemory bestStep = problem.getLastStep(11 + path); //get best step best prey
        problem.setLastStep(obj_function_num[count_best_prey], bestStep); //save best step as last step best prey
        current_obj_function_values[obj_function_num[count_best_prey]] = problem.getFunctionValue(obj_function_num[count_best_prey]);
        System.out.println("best prey no." + obj_function_num[count_best_prey] + " make movement +" + i + " become " + llh_obj_function_values[path]);

        LastStepMemory lastStep = problem.getLastStep(obj_function_num[count_best_prey]);
        System.out.println("best prey no." + obj_function_num[count_best_prey] + " move with (" + lastStep.getLlh_type() + ", " + lastStep.param1 + ", "
                + lastStep.param2 + ", " + lastStep.param3 + ")");
    } else {
        int path = find_bestPath(llh_obj_function_values, 0, llh_obj_function_values.length);
        temp_obj_func = problem.getFunctionValue(11 + path);
        if (temp_obj_func != Double.MAX_VALUE) {
            //Simulated Annealing
            double delta = current_obj_function_values[obj_function_num[count_best_prey]] - temp_obj_func;
            double ret = Math.pow(-(delta / T_SA), 2);
            double random = ThreadLocalRandom.current().nextDouble(0, 1);
            if (ret > random) { //accept
                problem.copySolutionRl(11 + path, obj_function_num[count_best_prey]);
                LastStepMemory bestStep = problem.getLastStep(11 + path); //get best step best prey
                problem.setLastStep(obj_function_num[count_best_prey], bestStep); //save best step as last step best prey
                current_obj_function_values[obj_function_num[count_best_prey]] = problem.getFunctionValue(obj_function_num[count_best_prey]);
                System.out.println("best prey no." + obj_function_num[count_best_prey] + " make movement SA +" + i + " become " + llh_obj_function_values[path]);
            } else {
                System.out.println("best prey no." + obj_function_num[count_best_prey] + " no movement SA +" + i);
            }
        } else {
            System.out.println("best prey no." + obj_function_num[count_best_prey] + " no movement +" + i);
        }
    }
}
```

Gambar 5. 9 Kode Pergerakan best prey pada Prey-Predator

Pada gambar 5.9 dijelaskan pergerakan *best prey* pada algoritma *Prey Predator*. Disini random direction akan dibuat, dimana perubahan solusi tersebut akan disimpan di dalam memory selain indeks satu dan indeks solusi. Kemudian

tiap solusi akan dihitung nilai fungsi obyektifnya, apabila nilai fungsi obyektif memiliki nilai yang lebih baik dari *best prey*, *direction* untuk *best prey* telah tersedia. Apabila dalam *random direction* tidak ada nilai obyektif yang lebih baik daripada *best prey*, maka *best prey* akan menjalankan *Simulated Annealing*. *Direction* yang memiliki nilai fungsi obyektif lebih baik daripada *best prey* kemudian dipilih nilai yang terbaik. Kemudian nilai *best prey* akan berubah sesuai dengan *direction* yang terbaik.

```
/////// ordinary prey
for (int count_ord_prey = best_prey; count_ord_prey < (ordinary_prey + best_prey); count_ord_prey++) { //until last ordinary prey
    Random r2 = new Random();
    double percentage = r2.nextDouble();
    if (percentage > pf) { // not get pf, moving randomly
        int llh_ordinary_prey = r2.nextInt(number_of_heuristics);
        problem.applyHeuristic(llh_ordinary_prey, obj_function_num[count_ord_prey], 11);
        temp_obj_func = problem.getFunctionValue(11);
        if (temp_obj_func != Double.MAX_VALUE) {
            problem.copySolutionR1(11, obj_function_num[count_ord_prey]);
            LastStepMemory lastStepOrd = problem.getLastStep(11); //get last step
            problem.setLastStep(obj_function_num[count_ord_prey], lastStepOrd); //save step as last step
            System.out.println("ord prey no." + obj_function_num[count_ord_prey] + " move randomly");
        } else {
            System.out.println("ord prey no." + obj_function_num[count_ord_prey] + " cannot move INFEASIBLE");
        }

    } else {// get pf follow better prey randomly

        int better_prey = best_prey + (count_ord_prey - best_prey);
        System.out.println("better prey is " + better_prey);
        if (better_prey > 1) {
            better_prey = ThreadLocalRandom.current().nextInt(0, better_prey - 1);
              better_prey = r2.nextInt(better_prey); //get randomly better prey
        }
        System.out.println("ord prey no." + obj_function_num[count_ord_prey] + " want follow better prey no." + obj_function_num[better_prey]);
        LastStepMemory lastStep2 = problem.getLastStep(obj_function_num[better_prey]); //get last step for prey
        if (lastStep2 != null) {
            problem.applyHeuristicWithLastStep(lastStep2, obj_function_num[count_ord_prey], 11);
            temp_obj_func = problem.getFunctionValue(11);
            if (temp_obj_func != Double.MAX_VALUE) {
                problem.copySolutionR1(11, obj_function_num[count_ord_prey]);
                problem.setLastStep(obj_function_num[count_ord_prey], lastStep2);
                System.out.println("ord prey no." + obj_function_num[count_ord_prey] + " follow better prey no." + obj_function_num[better_prey]);
            } else {
                System.out.println("ord prey no." + obj_function_num[count_ord_prey] + " cannot move INFEASIBLE");
            }
        } else {
            System.out.println("ord prey no." + obj_function_num[count_ord_prey] + " cannot follow prey no." + obj_function_num[better_prey] + "doesn't have movement")
        }

    }
}
```

Gambar 5. 10 Kode pergerakan ordinary prey

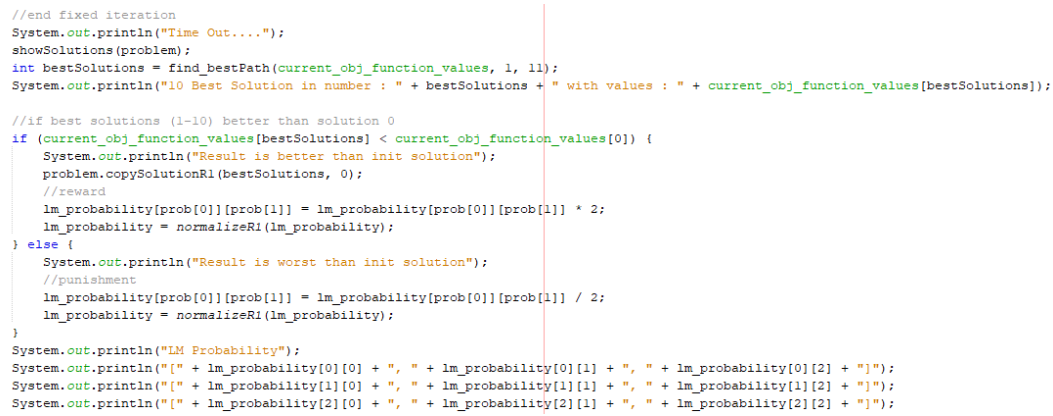
Pada Gambar 5.10 dijelaskan pergerakan *prey* pada algoritma *Prey Predator*. Disini akan dihitung nilai random, apabila nilai random masuk ke dalam pf atau *probability of follow*, maka ordinary prey akan mengikuti *best prey*. Apabila nilai random berada di luar atau di atas pf, maka *ordinary prey* akan melakukan *Simple Random* dan menerima apapun hasil dari algoritma *Simple Random*.

```
//////predator
for (int count_predator = best_prey + ordinary_prey; count_predator < (ordinary_prey + best_prey + predator); count_predator++) {
    Random r2 = new Random();
    double percentage = r2.nextDouble();
    if (percentage > pf) { // not get pf, moving randomly
        int llh_ordinary_prey = r2.nextInt(number_of_heuristics);
        problem.applyHeuristic(llh_ordinary_prey, obj_function_num[count_predator], 11);
        temp_obj_func = problem.getFunctionValue(11);
        if (temp_obj_func != Double.MAX_VALUE) {
            problem.copySolutionR1(11, obj_function_num[count_predator]);
            System.out.println("predator no." + obj_function_num[count_predator] + " move randomly");
        } else {
            System.out.println("predator no." + obj_function_num[count_predator] + " cannot move INFEASIBLE");
        }
    } else {// get pf follow prey randomly
        int prey = ThreadLocalRandom.current().nextInt((best_prey + 1), (ordinary_prey + best_prey));
        LastStepMemory lastStep3 = problem.getLastStep(obj_function_num[prey]); //get last step for prey
        if (lastStep3 != null) {
            problem.applyHeuristicWithLastStep(lastStep3, obj_function_num[count_predator], 11);
            temp_obj_func = problem.getFunctionValue(11);
            if (temp_obj_func != Double.MAX_VALUE) {
                problem.copySolutionR1(11, obj_function_num[count_predator]);
                System.out.println("predator no." + obj_function_num[count_predator] + " follow prey no." + obj_function_num[prey]);
            } else {
                System.out.println("predator no." + obj_function_num[count_predator] + " cannot move INFEASIBLE");
            }
        } else {
            System.out.println("predator no." + obj_function_num[count_predator] + " cannot follow prey no." + obj_function_num[prey] + "doesn't have movement");
        }

    }
}
//end movement predator
```

Gambar 5. 11 Kode pergerakan predator

Pada Gambar 5.11 dijelaskan pengkodean pergerakan predator pada algoritma Prey-Predator. Hal pertama yang dilakukan adalah menghitung nilai random, apabila nilai random berada di bawah probability of follow, maka predator akan mengikuti *ordinary prey*, akan tetapi apabila nilai random berada di atas *probability of follow*, maka predator akan melakukan pergerakan acak berupa algoritma *Simple Random*.



```
//end fixed iteration
System.out.println("Time Out....");
showSolutions(problem);
int bestSolutions = find_bestPath(current_obj_function_values, 1, 11);
System.out.println("10 Best Solution in number : " + bestSolutions + " with values : " + current_obj_function_values[bestSolutions]);

//if best solutions (1-10) better than solution 0
if (current_obj_function_values[bestSolutions] < current_obj_function_values[0]) {
    System.out.println("Result is better than init solution");
    problem.copySolutionR1(bestSolutions, 0);
    //reward
    lm_probability[prob[0]][prob[1]] = lm_probability[prob[0]][prob[1]] * 2;
    lm_probability = normalizeR1(lm_probability);
} else {
    System.out.println("Result is worst than init solution");
    //punishment
    lm_probability[prob[0]][prob[1]] = lm_probability[prob[0]][prob[1]] / 2;
    lm_probability = normalizeR1(lm_probability);
}
System.out.println("LM Probability");
System.out.println("[" + lm_probability[0][0] + ", " + lm_probability[0][1] + ", " + lm_probability[0][2] + "]");
System.out.println("[" + lm_probability[1][0] + ", " + lm_probability[1][1] + ", " + lm_probability[1][2] + "]");
System.out.println("[" + lm_probability[2][0] + ", " + lm_probability[2][1] + ", " + lm_probability[2][2] + "]");
```

Gambar 5. 12 Kode normalisasi Prey-Predator

Pada Gambar 5.12 dijelaskan pengkodean *reward* dan *punishment* pada Prey-Predator. Setelah perulangan pada *best prey*, *ordinary prey*, dan *predator* selesai, maka tahap selanjutnya adalah mencari solusi terbaik di antara solusi yang ada. Kemudian solusi terbaik tersebut dibandingkan dengan solusi awal, apabila

solusi dari perulangan tersebut lebih baik dari solusi awal, maka rasio yang berada pada *learning matrix* akan diberikan reward berupa dikali dua dan solusi awal digantikan oleh solusi terbaik, apabila solusi terbaik yang dihasilkan memiliki nilai fungsi obyektif yang lebih buruk dari solusi awal, maka rasio pada *learning matrix* akan diberi *punishment* berupa dibagi dua. Kemudian *learning matrix* akan di normalisasikan akan jumlah dari *learning matrix* kurang dari satu. Kemudian proses tersebut akan diulang kembali hingga *time limit* habis.

# BAB 6
# HASIL IMPLEMENTASI DAN ANALISIS

Dalam tahap ini ditampilkan hasil implementasi dari strategi Prey-Predator dan Simulated Annealing (PPA-SA) pada hyperheuristic (HH) yang diujikan terhadap sebuah problem domain seperti yang telah dijelaskan pada sebelumnya. Setelah itu, dilakukan evaluasi kinerja terhadap strategi tersebut.

Pengukuran kinerja dilakukan dengan cara membandingkan hasil uji coba metode yang diusulkan dengan metode lainnya. Detail dari hasil implementasi dan analisis hasil dijelaskan pada sub bab di bawah ini.

.

## 6.1.Hasil Implementasi PPA-SA-HH pada Dataset ITC 2007

Data detail hasil eksekusi 10 kali terhadap PPA-SA-HH untuk instance dari dataset ITC 2007, dalam batas waktu yang telah ditentukan. Tabel 6.1 merupakan data hasil perhitungan nilai terbaik (minimum) dan rata-rata dari nilai fungsi fitness pada setiap instance domain permasalahan.

*Tabel 6. 1 Hasil Eksperimen PPA-SA*

| ProblemInstance | PPA-SA ITC 2007 | |
|---|---|---|
| | **Best** | **Average** |
| Exam1 | 28789 | 30668 |
| Exam2 | 29677 | 35970 |
| Exam3 | 80596 | 96916 |
| Exam4 | - | - |
| Exam5 | 82225 | 92530 |
| Exam6 | 47240 | 52125 |
| Exam7 | 55386 | 60173 |
| Exam8 | 84084 | 101094 |

Tabel 6.2 Hasil Eksperimen PPA-SA Pada Dataset ITC 2007 Dibandingkan Hasil Eksperimen sebelumnya yang mengusulkan algoritma yang berbeda

*Tabel 6. 2 Perbandingan hasil PPA-SA dengan SR-SA, SR-HC, dan PPA*

| | Solusi Awal | SR-HC-HH | SR-SA-HH | PPA-HH | PPA-SA-HH |
|---|---|---|---|---|---|
| **EXAM 1** | 31111 | 31154 | 30793 | 30961 | 30668 |
| **EXAM 2** | 39883 | 39322 | 38947 | 40253 | 35970 |
| **EXAM 3** | 104385 | 102826 | 98906 | 100291 | 96916 |
| **EXAM 4** | - | - | - | - | - |
| **EXAM 5** | 139994 | 129837 | 113602 | 92459 | 92530 |
| **EXAM 6** | 54267 | 55106 | 51858 | 54069 | 52125 |
| **EXAM 7** | 66522 | 67890 | 62897 | 65308 | 60173 |
| **EXAM 8** | 132811 | 129388 | 119172 | 99098 | 101094 |

Pada Tabel 6.2 terlihat peningkatan hasil menjadi lebih baik dari algoritma Prey-Predator dan Simulated Annealing apabila dibandingkan dengan algoritma yang lainnya Simple Random – Hill Climbing dan Simplre Random – Simulated Annealing. Selain pada instance 5, 6, dan 8 algoritma PPA-SA mampu mengungguli algoritma percobaan yang lainnya. Algoritma SR-SA sendiri mampu memiliki nilai terbaik pada instance 6 dan algoritma PPA mampu mengungguli algoritma lainnya pada exam 5 dan 8.

Pada Tabel 6.2 hasil dari penelitian ini terlihat bahwa exam 4 tidak memiliki nilai karena hingga waktu yang ditentukan solusi awal tidak dapat dibentuk, oleh karena itu program akan dihentikan paksa. Hal ini dapat terjadi karena kompleksitas dari data tersebut atau tingginya *concflict density*. Pada instance 4 terdapat 21 period, 273 exam, 5045 student, dan hanya satu *room*.

Pada awalnya exam 1 memiliki masalah yang sama, akan tetapi masalah ini dihadapi dengan perubahan pada algoritma. Dimana apabila pembentukan solusi awal gagal atau memakan waktu yang cukup lama, maka pembentukan solusi awal akan di ulang hingga terbentuk solusi awal yang menghasilkan nilai yang *feasible*. Hal ini sendiri menyebabkan pembentukan solusi pada exam 1 membutuhkan waktu yang lebih lama daripada exam yang lain.

*Tabel 6. 3 Pelanggaran pada algoritma PPA-SA*

| Exam | Cost | TIR | TID | PS | NMD | FL | SP | SR |
|---|---|---|---|---|---|---|---|---|
| 1 | 30668 | 7434 | 0 | 10808 | 1827 | 276 | 2513 | 7810 |
| 2 | 35970 | 19839 | 5012 | 2061 | 3380 | 925 | 3575 | 1178 |
| 3 | 96916 | 45570 | 16557 | 19895 | 9718 | 1108 | 800 | 3268 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 92530 | 32580 | 12246 | 11732 | 0 | 1802 | 34170 | 0 |
| 6 | 52124 | 23338 | 0 | 19900 | 3377 | 375 | 899 | 4235 |
| 7 | 60173 | 14715 | 0 | 16803 | 9243 | 937 | 9355 | 9120 |
| 8 | 101080 | 52065 | 0 | 19335 | 25997 | 460 | 520 | 2703 |

Keterangan:
TIR: Two exam in a Row
TID: Two exam in a Day
PS: Period Spread
NMD: Non-Mixed Duration
FL: Front Load
SP: Soft Period Penalty
SR: Soft Room Penalty

Pada tabel 6.3 terlihat pelanggaran yang dilakukan oleh algoritma kombinasi Prey-Predator dengan Simulated Annealing. Pada exam 1 memiliki pelanggaran terbanyak pada *period spread,* pinalti ini diberikan saat murid harus melakukan lebih dari satu ujian dalam jangka waktu periode yang ditentukan, dimana dalam exam 1 jarak yang ditentukan adalah 5. Pada exam 1 menunjukkan bahwa pelanggaran yang dilakukan adalah banyak murid yang melakukan ujian tanpa diberi waktu istirahat sebanyak 5 *period*. Hal ini menunjukkan bahwa algoritma kesultan untuk mencari waktu istirahat yang diberikan untuk murid pada exam 1 dan 7.

Pada exam 2 pelanggaran yang paling banyak dilakukan adalah *two exam in a row*, dimana pelanggaran ini dihitung apabila mahasiswa melakukan ujian lebih dari satu kali dengan periode yang berurutan pada hari yang sama. Nilai pelanggaran *two exam in a row* pada exam 2 adalah 15. Hal ini menunjukkan bahwa algoritma pada exam 2 kesulitan untuk mencari waktu pada murid agar dapat beristirahat setelah ujian selesai.

Pada exam 3 pelanggaran yang dilakukan bukan hanya pada two exam in a row, pelanggaran *two exam in a day* dan *period spread* memiliki nilai yang cukup banyak, sedangkan jumlah *time slot*, ujian, dan *room* pada exam 3 hampir sama dengan exam 2, hal ini dikarenakan jumlah murid yang lebih banyak dari pada exam 2. Hal ini menunjukkan algoritma pada exam 3 bukan hanya kesulitan untuk mencari waktu pada murid agar dapat beristirahat setelah ujian selesai.

Pada exam 5 pelanggaran yang paling banyak dilakukan adalah *soft period penalty*, dimana pelanggaran ini dihitung setiap *period* ini digunakan. Pada exam 5 terdapat 16 dari 42 period yang memiliki yang memiliki nilai pelanggaran. Nilai *soft period penalty* yang terbesar adalah 1000. Hal ini menunjukkan bahwa algoritma sering menggunakan periode yang memiliki nilai pinalti atau pelanggaran yang besar. Pelanggaran terbesar kedua pada exam 5 adalah *two exam in a row*, hal tersebut dikarenakan pada exam 5 *two exam in row* memiliki nilai pelanggaran yang cukup tinggi yaitu 40. Hal ini menunjukkan bahwa algoritma pada exam 5 kesulitan untuk mencari waktu pada murid agar dapat beristirahat setelah ujian selesai dan terlalu banyak menggunakan periode yang memiliki nilai pelanggaran yang besar.

Pada exam 6 pelanggaran yang paling banyak dilakukan adalah two exam in a row dan period spread. Pada exam 6 nilai pelanggaran pada two exam in a row adalah 20, sedangkan nilai dari period spread adalah 20. Nilai pelanggaran period spread pada exam 6 merupakan nilai yang terbesar apabila dibandingkan dengan exam yang lain. Pelanggaran tersebut menunjukkan bahwa algoritma pada exam 6 kesulitan untuk mencari waktu pada murid agar dapat beristirahat setelah ujian selesai.

Pada exam 7 pelanggaran yang paling banyak dilakukan adalah period spread dan two exam in a row. Nilai pada period spread adalah 10 sedangkan nilai pelanggaran pada two exam in a row adalah 25, hal ini menunjukkan bahwa algoritma pada exam 7 kesulitan untuk mencari waktu istirahat pada murid setelah ujian.

Pada exam 8 pelanggaran yang paling banyak dilakukan adalah two exam in a row dan non-mixed duration. Nilai pelanggaran non-mixed duration diberikan apabila ada beberapa ujian menggunakan ruang ujian tetapi memiliki durasi ujian yang berbeda. Nilai pelanggaran pada two exam in a row dan non-mixed duration pada exam 8 memiliki nilai pelanggaran yang lebih besar dibandingkan dengan exam lainnya. Hal ini menunjukkan bahwa algoritma pada exam 8 kesulitan untuk mencari waktu istirahat pada murid setelah ujian selesai dan ujian yang sama memiliki durasi waktu yang berbeda.

*Halaman ini sengaja dikosongkan*

# BAB 7
# KESIMPULAN DAN SARAN

Bab ini menjelaskan kesimpulan yang diambil berdasarkan hasil uji coba yang telah dilakukan. Selain itu, dalam bab ini juga diberikan saran penegembangan untuk penelitian dimasa yang akan datang.

## 7.1.Kesimpulan

. Pada penelitian ini, diperkenalkan kombinasi antara algoritma Prey-Predator sebagai seleksi low-level heuristic dan Simulated Annealing sebagai move acceptance pada permasalahan Examination Timetabling sehingga memenuhi kriteria hyper-heuristic yang memiliki penyeleksian LLH dan move acceptance. Kesimpulan dari penggabungan algoritma sebagai berikut:

a) Penggabungan algoritma Prey-Predator dengan Simulated Annealing berhasil dilakukan dengan memenuhi seluruh hard constraint pada permasalahan examination timetabling dengan dataset ITC 2007
b) Hasil eksperimen pada dataset ITC 2007 dengan menggunakan algoritma *Prey-Predator* dan *Simulated Annealing* menunjukkan bahwa hasil algoritma tersebut lebih baik daripada menggunakan algoritma *Simple Random – Hill Climbing, Simple Random – Simulated Annealing,* dan *Prey-Predator* kecuali pada instance 6 dimana Simple Random – Simulated Annealing memiliki nilai yang lebih baik daripada yang lain dan instance 5 dan 8 oleh algoritma Prey-*Predator*. Akan tetapi hasil dari algoritma yang diusulkan tidak lebih baik daripada penelitian-penelitian sebelumnya, terutama pada instance 4 dimana pembentukan solusi awal tidak bisa dilakukan dan memakan waktu yang cukup lama sehingga nilai bersifat 0.
c) Penggabungan algoritma Prey-Predator menggunakan Simulated Annealing menunjukkan hasil yang baik daripada hanya menggunakan *Prey-Predator*. Hal ini menunjukkan bahwa *Simulated Annealing* berhasil mengembangkan *best prey* pada *Prey-Predator* agar tidak terjebak dalam *local optimum*, sehingga menghasilkan nilai yang lebih baik.

## 7.2.Saran

Dataset ITC 2007 memiliki persoalan yang kompleks. Kompleksitas tersebut ditunjukkan dengan banyaknya data yang harus ditempatkan pada sebuah penjadwalan, dimana hard constraint dan soft constraint harus dipertimbangkan pada saat proses penjadwalan.

Pada penggabungan algoritma Prey-Predator sebagai seleksi LLH dan Simulated Annealing sebagai move acceptance, memiliki solusi yang cukup baik apabila dibandingkan dengan beberapa algoritma percobaan seperti SR-HC, SR-SA, dan Prey-Predator. Hal ini menunujukkan bahwa Simulated Annealing berhasil meningkatkan performa Prey-Predator meskipun hanya diterapkan pada best prey. Hasil pada penelitian ini belum menunjukkan hasil yang baik apabila dibandingkan dengan penelitian-penelitian yang telah dilakukan sebelumnya. Sehingga diharapkan untuk penelitian selanjutnya, dapat dilakukan pengujian dengan modifikasi sebagai berikut:

a) Menggunakan strategi seleksi LLH yang sama Prey-Predator dan menggantinya dengan move acceptance yang lain, seperti Step Counting Hill Climbing dan Late Acceptance Hill Climbing dimana algoritma tersebut tidak terjebak dalam local optimum, akan tetapi masih mempertahankan perbedaan dengan solusi awal.
b) Menggunakan low-level-heurisitic (LLH) yang lain, selain yang digunakan pada penelitian ini, agar mengetaahui hasil yang diberikan

# DAFTAR PUSTAKA


H. Bustani, *Fundamental Operation Research*. Gramedia Pustaka Utama.

M. Carter, C. C. Price, and G. Rabadi, *Operations Research: A Practical Introduction*. CRC Press, 2018.

Di Wang and Jiuping Xu, "A fuzzy multi-objective optimizing scheduling for operation room in hospital," in *2008 IEEE International Conference on Industrial Engineering and Engineering Management*, Dec. 2008, pp. 614–618, doi: 10.1109/IEEM.2008.4737942.

M. Hsie, W. Hsiao, T. Cheng, and H. Chen, "A model used in creating a work-rest schedule for laborers," *Automation in Construction*, vol. 18, no. 6, pp. 762–769, Oct. 2009, doi: 10.1016/j.autcon.2009.02.010.

Baker, K.R., 1974. Introduction To Sequencing and Scheduling, Jhon Willey and Sons, Inc. New York

Baker, K.R., Trietsch, D. Principles Of Sequencing And Scheduling.

Doulaty, M., Derakhshi, M.R.F., Abdi, M., Timetabling: A State-of-the-Art Evolutionary Approach. International Journal of Machine Learning and Computing. 3, 255

R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *J Sched*, vol. 12, no. 1, pp. 55–89, Feb. 2009, doi: 10.1007/s10951-008-0077-5.

A. Schaerf, "A Survey of Automated Timetabling," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 87–127, Apr. 1999, doi: 10.1023/A:1006576209967.

S. Even, A. Itai, and A. Shamir, "On the complexity of time table and multi-commodity flow problems," in *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, Oct. 1975, pp. 184–193, doi: 10.1109/SFCS.1975.21.

M. W. Carter, G. Laporte, and S. Y. Lee, "Examination Timetabling: Algorithmic Strategies and Applications," *J Oper Res Soc*, vol. 47, no. 3, pp. 373–383, Mar. 1996, doi: 10.1057/jors.1996.37.

L. Di Gaspero, B. Mccollum, and A. Schaerf, "The Second International Timetabling Competition (ITC-2007): Curriculum-based Course Timetabling (Track 3)," Jan. 2007.

G. Laporte and S. Desroches, "Examination timetabling by computer," *Computers & Operations Research*, vol. 11, no. 4, pp. 351–360, Jan. 1984, doi: 10.1016/0305-0548(84)90036-4.

R. W. Eglese and G. K. Rand, "Conference Seminar Timetabling," *Journal of the Operational Research Society*, vol. 38, no. 7, pp. 591–598, Jul. 1987, doi: 10.1057/jors.1987.102.

D. Corne, H. Fang, and C. Mellish, "Solving the Modular Exam Scheduling Problem with Genetic Algorithms," Jan. 1993.

R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination

timetabling," *J Sched*, vol. 12, no. 1, pp. 55–89, Feb. 2009, doi: 10.1007/s10951-008-0077-5.

P. Cowling, G. Kendall, and E. Soubeiga, "A Hyperheuristic Approach to Scheduling a Sales Summit," in *Practice and Theory of Automated Timetabling III*, Berlin, Heidelberg, 2001, pp. 176–190, doi: 10.1007/3-540-44629-X_11.

P. Demeester, B. Bilgin, P. De Causmaecker, and G. Vanden Berghe, "A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice," *J Sched*, vol. 15, no. 1, pp. 83–103, Feb. 2012, doi: 10.1007/s10951-011-0258-5.

C. Gogos, P. Alefragis, and E. Housos, "An improved multi-staged algorithmic process for the solution of the examination timetabling problem," *Annals OR*, vol. 194, pp. 203–221, Apr. 2012, doi: 10.1007/s10479-010-0712-3.

S. L. Tilahun, "Prey-predator algorithm for discrete problems: a case for examination timetabling problem," *Turk J Elec Eng & Comp Sci*, vol. 27, no. 2, pp. 950–960, Mar. 2019.

S. L. Tilahun, "Prey predator hyperheuristic," *Applied Soft Computing*, vol. 59, pp. 104–114, Oct. 2017, doi: 10.1016/j.asoc.2017.04.044.

A. L. Bolaji, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "A hybrid nature-inspired artificial bee colony algorithm for uncapacitated examination timetabling problems," *Journal of Intelligent Systems*, vol. 24, no. 1, pp. 37–54, 2015.

Y. Bykov and S. Petrovic, "A step counting hill climbing algorithm applied to university examination timetabling," *Journal of Scheduling*, vol. 19, no. 4, pp. 479–492, 2016.

M. Cheraitia and S. Haddadi, "Simulated annealing for the uncapacitated exam scheduling problem," *International Journal of Metaheuristics*, vol. 5, no. 2, pp. 156–170, 2016.

A. K. Mandal and M. N. M. Kahar, "Solving examination timetabling problem using partial exam assignment with great deluge algorithm," in *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, 2015, pp. 530–534.

A. K. Mandal and M. N. M. Kahar, "Solving examination timetabling problem using partial exam assignment with hill climbing search," in *2015 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, 2015, pp. 84–89.

A. Mukhlason, "Hyper-heuristics and fairness in examination timetabling problems," Jul. 2017.

B. Bahmani-Firouzi, S. Sharifinia, R. Azizipanah-Abarghooee, and T. Niknam, "Scenario-based optimal bidding strategies of GENCOs in the incomplete information electricity market using a new improved prey—predator optimization algorithm," *IEEE Systems Journal*, vol. 9, no. 4, pp. 1485–1495, 2015.

N. Hamadneh, W. Khan, and S. Tilahun, "Optimization of microchannel heat sinks using prey-predator algorithm and artificial neural networks," *Machines*, vol. 6, no. 2, p. 26, 2018.

S. Tilahun, S. Sathasivam, H. C. Ong, and N. Hamadneh, “Prey-Predator Algrorithm as a New Optimization Technique Using in Radia Basis Function Neural Networks,” *Research Journal of Applied Sciences*, vol. 7, pp. 383–387, Oct. 2013, doi: 10.3923/rjasci.2013.383.387.

E. K. Burke and S. Petrovic, “Recent research directions in automated timetabling,” *European Journal of Operational Research*, vol. 140, no. 2, pp. 266–280, Jul. 2002, doi: 10.1016/S0377-2217(02)00069-3.

H. Turabieh and S. Abdullah, “An integrated hybrid approach to the examination timetabling problem,” *Omega*, vol. 39, no. 6, pp. 598–607, Dec. 2011, doi: 10.1016/j.omega.2010.12.005.

N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, “A Dynamic Multiarmed Bandit-Gene Expression Programming Hyper-Heuristic for Combinatorial Optimization Problems,” *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 217–228, Feb. 2015, doi: 10.1109/TCYB.2014.2323936.

A. P. Dornelles, “A matheuristic approach for solving the high school timetabling problem,” 2015.

B. Mccollum, P. McMullan, A. Parkes, E. K. Burke, and R. Qu, “A New Model for Automated Examination Timetabling,” *Annals of Operations Research*, vol. 194, pp. 291–315, Jan. 2012.

E. Burke, M. R. Hyde, G. Kendall, G. Ochoa, and E. Özcan, “A classification of hyper-heuristic approaches,” in *International Series in Operations Research and Management Science*, 2010, pp. 449–468.

P. Cowling, G. Kendall, and E. Soubeiga, “A Hyperheuristic Approach to Scheduling a Sales Summit,” in *Practice and Theory of Automated Timetabling III*, Berlin, Heidelberg, 2001, pp. 176–190, doi: 10.1007/3-540-44629-X_11.

A. S. Ferreira, “A cross-domain multi-armed bandit hyper-heuristic,” 2016, Accessed: Jul. 30, 2020. [Online]. Available: https://acervodigital.ufpr.br/handle/1884/41803.

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, May 1983, doi: 10.1126/science.220.4598.671.

T. Müller, “ITC2007 solver description: A hybrid approach,” *Annals of Operations Research*, vol. 172, pp. 429–446, Jan. 2008, doi: 10.1007/s10479-009-0644-y.

S. Abdul Rahman, A. Bargiela, E. K. Burke, E. Özcan, B. McCollum, and P. McMullan, “Adaptive linear combination of heuristic orderings in constructing examination timetables,” *European Journal of Operational Research*, vol. 232, no. 2, pp. 287–297, Jan. 2014, doi: 10.1016/j.ejor.2013.06.052.

M. Alzaqebah and S. Abdullah, “An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling,” *J Sched*, vol. 17, no. 3, pp. 249–262, Jun. 2014, doi: 10.1007/s10951-013-0352-y.

E. K. Burke, R. Qu, and A. Soghier, “Adaptive selection of heuristics for improving exam timetables,” *Ann Oper Res*, vol. 218, no. 1, pp. 129–145, Jul. 2014, doi: 10.1007/s10479-012-1140-3.

# DAFTAR LAMPIRAN

Lampiran A. Hasil Pengujian PPA-SA-HH pada dataset ITC 2007

| No. | INSTANCE 1 | INSTANCE 2 | INSTANCE 3 | INSTANCE 4 | INSTANCE 5 | INSTANCE 6 | INSTANCE 7 | INSTANCE 8 |
|---|---|---|---|---|---|---|---|---|
| | **EXAM 1** | **EXAM 2** | **EXAM 3** | **EXAM 4** | **EXAM 5** | **EXAM 6** | **EXAM 7** | **EXAM 8** |
| 1 | 30053 | 41365 | 100095 | NA | 104705 | 49150 | 58575 | 89425 |
| 2 | 28789 | 40087 | 103087 | NA | 89393 | 57515 | 57971 | 91440 |
| 3 | 30234 | 30695 | 94275 | NA | 88004 | 49190 | 58734 | 91820 |
| 4 | 29571 | 33323 | 87582 | NA | 82645 | 50765 | 59166 | 124590 |
| 5 | 30452 | 40397 | 85540 | NA | 94845 | 47240 | 66418 | 98132 |
| 6 | 30430 | 40301 | 97825 | NA | 95269 | 57865 | 59057 | 103131 |
| 7 | 32296 | 29677 | 119855 | NA | 82225 | 50205 | 67234 | 99058 |
| 8 | 31389 | 34023 | 104653 | NA | 96928 | 53090 | 55386 | 126246 |
| 9 | 32770 | 30770 | 80596 | NA | 97867 | 47360 | 58495 | 103105 |
| 10 | 30704 | 38888 | 95656 | NA | 93426 | 58870 | 60697 | 84084 |
| | | | | | | | | |
| **AVG** | **30668** | **35970** | **96916** | NA | **92530** | **52125** | **60173** | **101094** |
| *BEST* | 28789 | *29677* | *80596* | NA | *82225* | *47240* | *55386* | *84084* |

# BIOGRAFI PENULIS

Rusdi Hamidan, lahir di Semarang pada tanggal 21 November 1991, merupakan anak ketiga dari tiga bersaudara. Penulis menempuh pendidikan formal di SD Medokan Ayu II Surabaya, SMPN 35 Surabaya, SMAN 3 Surabaya. Penulis melanjutkan S1 di Jurusan Teknik Informatika Universitas Dr. Soetomo Surabaya (Unitomo) pada tahun 2010 dan menyelesaikan pendidikan tahun 2014. Semasa menempuh pendidikan S1, penulis aktif sebagai asisten laboratorium pada tahun 2010-2011. Pada tahun 2013 penulis mengikuti lomba KRPAI Beroda Regional IV dan berhasil meraih juara I. Pada tahun 2016 penulis melanjutkan kuliah ke jenjang S2 pada program studi Sistem Informasi Institut Teknologi Sepuluh Nopember (ITS). Pada tahun 2018, penulis diangkat menjadi Kepala UPT Komputer di Universitas Dr. Soetomo Surabaya. Untuk tema penelitian, penulis bergabung dengan tim laboratorium Rekayasa Data dan Intelegensi Bisnis (RDIB). Penulis menyelesaikan studi S2 pada tahun 2020. Kritik dan saran yang membangun dapat dikirim ke email rusdi.hamidan@gmail.com.