BAB III
PENYELESAIAN MASALAH PENJADWALAN UJIAN DENGAN MENGGUNAKAN METODE *GREAT DELUGE HYPER-HEURISTIC*

Bab ini membahas mengenai masalah penjadwalan ujian, metodologi penelitian, model optimisasi, dan teknik penyelesaian model optimisasi penjadwalan ujian menggunakan metode *Great Deluge Hyper-Heuristic*.

3.1. Deskripsi Masalah

Penjadwalan ujian merupakan hal yang penting bagi bidang pendidikan. Masalah penjadwalan ujian adalah masalah optimasi kombinatorial dunia nyata yang sulit dipecahkan karena memiliki banyak kendala dan sumber daya yang terbatas (yaitu, slot waktu dan ruangan) dalam mengalokasikan sejumlah besar ujian (Mandal, Kahar, & Kendall, 2020). Dalam skala yang kecil, penjadwalan ujian dapat dilakukan secara manual. Akan tetapi, untuk skala yang lebih besar dan kompleks, jika penjadwalan dilakukan manual akan membutuhkan waktu yang lama, dan beresiko terjadinya bentrok jadwal dikarenakan banyaknya hal yang perlu diperhatikan seperti memastikan bahwa setiap mahasiswa tidak memiliki lebih dari satu jadwal ujian pada slot waktu yang sama, memperhatikan jumlah mahasiswa tidak melebihi kapasitas ruang ujian (Icasia, Tyasnurita, & Purba, 2020). Jumlah ujian yang diikuti mahasiswa juga perlu diperhatikan, karena semakin banyak jumlah ujian yang diikuti oleh satu mahasiswa dalam satu hari juga dapat berimbas nyata pada hasil ujian yang didapat. Dalam kasus mahasiswa yang mengikuti lebih dari satu ujian pada hari yang sama, jeda antar ujian juga menjadi pertimbangan penting agar mahasiswa memiliki waktu cukup untuk belajar atau istirahat sebelum ujian selanjutnya dilaksanakan. Oleh karena itu, kompleksitas penjadwalan yang rumit dengan beberapa komponen yang memiliki keterkaitan menjadikan proses pembuatan jadwal ujian bukanlah hal yang mudah dan membutuhkan waktu yang tidak sedikit (Syahrani, 2018).

Masalah penjadwalan ujian dapat didefinisikan sebagai alokasi ujian ke dalam satu slot waktu yang dapat ditempati ketika memenuhi *constraints* pada saat yang bersamaan (Supoyo & Azthanty, 2019). Terdapat dua kategori *constraint* yang menjadi batasan dalan penjadwalan ini yaitu *hard constraint* dan *soft*

constraint. *Hard constraint* merupakan batasan wajib yang mutlak untuk dipenuhi dan menjadi peraturan atau standar. *Hard constraint* pada penelitian ini mencakup:

1. Semua ujian harus dijadwalkan.
2. Tidak ada jadwal yang bentrok antara dua mata kuliah yang diujiankan yang diambil oleh mahasiswa yang sama.
3. Jumlah mahasiswa peserta ujian tidak melebihi kapasitas total ruang ujian dalam satu sesi.

Sedangkan *soft constraint* merupakan batasan yang tidak harus dipenuhi namun sangat disarankan dan diusahakan untuk dapat dipenuhi. *Soft constraint* dalam penelitian ini adalah:

1. Setiap mahasiswa tidak mengikuti dua ujian berturut-turut dalam sehari.
2. Setiap mahasiswa tidak mengikuti lebih dari dua ujian dalam sehari.

Tujuan dari penyelesaian penjadwalan ujian adalah untuk mendapat jadwal ujian yang optimal, yaitu dapat memenuhi *hard constraints* yang telah ditetapkan dan meminimumkan pelanggaran *soft constraint*. Dengan jadwal ujian yang optimal, diharapkan dapat berkorelasi positif dengan hasil capaian mahasiswa dalam ujian. Pada penelitian ini, masalah penjadwalan ujian akan diselesaikan menggunakan Algoritma *Great Deluge Hyper-Heuristic*.

3.2. Metodologi Penelitian

Tujuan dari penelitian ini adalah menyelesaikan masalah penjadwalan ujian di York Mills Collegiate Institute menggunakan metode *Great Deluge Hyper-Heuristic*. Tahapan penelitian diawali dengan mengidentifikasi masalah dan pembangunan model dari masalah penjadwalan ujian. Selanjutnya model akan diselesaikan menggunakan metode *Graph Colouring* untuk menghasilkan solusi awal yang kemudian akan dilanjutkan oleh *Great Deluge Hyper-Heuristic* untuk menemukan solusi yang lebih optimal. Berikut adalah tahapan penelitian yang dilakukan:

1. Identifikasi Masalah

   Tahap identifikasi permasalahan di salah satu mengenai penjadwalan ujian. Pada tahap ini dilakukan pengumpulan data, dan menentukan tujuan serta batasan penelitian yang dikerjakan, serta studi literatur mengenai penelitian-

penelitian sebelumnya dan pendekatan-pendekatan yang diperlukan untuk memecahkan permasalahan penjadwalan ujian.

2. Model Optimisasi

   Pada tahap ini dibangun model optimisasi masalah penjadwalan ujian di salah satu universitas dengan menetukan asumsi, mendefinisikan himpunan dan parameter, serta menentukan variabel keputusan, fungsi tujuan, dan fungsi kendala.

3. Teknik Penyelesaian Model

   Pada tahap ini model optimisasi penjadwalan ujian diselesaikan menggunakan algoritma *Great Deluge Hyper-Heuristic*. Adapun tahapan utama dari algoritma tersebut terdiri dari pembentukan solusi awal menggunakan *Graph Colouring* dan perbaikan solusi dengan Algoritma *Great Deluge Hyper-Heuristic*. Algoritma *Graph Colouring* digunakan untuk mencari solusi awal yang nantinya akan menjadi masukan untuk optimasi yang dilakukan algoritma *Great Deluge Hyper-Heuristic*. Di dalam penelitian ini, penulis menggunakan teknik *heuristic Largest Enrollment*, yaitu dengan menjadwalkan terlebih dahulu mata kuliah yang diujiankan, yang diambil oleh mahasiswa terbanyak. Jadwal yang telah dihasilkan dari algoritma *Graph Colouring* akan diperbaiki menggunakan *Great Deluge Hyper-Heuristic*.

4. Validasi

   Validasi dilakukan untuk menguji apakah model optimisasi dan metode *Great Deluge Hyper-Heuristic* sudah dapat diimplementasikan ke masalah *real* atau belum. Pada penelitian ini, validasi dilakukan dengan cara membandingkan solusi optimal yang dihasilkan oleh program *software* Python 3.10 dengan solusi optimal hasil perhitungan manual. Jika program dan perhitungan manual memiliki hasil yang sama, maka model optimisasi dan metode yang diusulkan dapat diimplementasikan pada masalah penjadwalan ujian di York Mills Collegiate Institute. Jika tidak, maka dapat dilakukan peninjauan kembali terhadap model yang didapatkan.

5. Implementasi

Pada tahap ini, dilakukan implementasi Algoritma *Great Deluge Hyper-Heuristic* pada penyusunan penjadwalan ujian di York Mills Collegiate Institute.

6. Analisis Hasil

Pada tahap ini, dilakukan analisis hasil terhadap penjadwalan dan performa algoritma *Great Deluge Hyper-Heuristic* dalam masalah penjadwalan ujian.

7. Kesimpulan

Pada tahap terakhir, dilakukan penarikan kesimpulan dari penggunaan metode *Great Deluge Hyper-Heuristic* untuk penjadwalan ujian.

3.3. Model Optimisasi Penjadwalan Ujian

Dalam tahap ini, akan dibangun model optimisasi dari massalah penjadwalan ujian. Ada lima langkah yang dilakukan untuk membuat pemodelan masalah secara matematis. Pertama, menentukan asumsi. Awal dari permodelan ini dimulai dengan menentukan asumsi yang berlaku dalam penjadwalan ujian. Asumsi yang diidentifikasi merupakan pra-kondisi yang dilakukan sebelum melakukan perhitungan atau penerapan algoritma. Kedua, mendefinisikan himpunan dan parameter. Pada tahap ini mendefinisikan semua himpunan dan parameter untuk memodelkan masalah penjadwalan ujian.

Ketiga, menentukan variabel keputusan. Pada tahap ini ditentukan variabel keputusan yang berlaku secara matematis. Variabel keputusan akan diidentifikasikan dalam bentuk biner 0-1. Keempat, menentukan fungsi tujuan. Dalam tahap ini ditentukan fungsi tujuan yang dalam penelitian ini adalah meminimumkan pelanggaran *soft constraint*. Kelima, menentukan *constraint* (batasan). Pada tahap ini ditetapkan batasan-batasan masalah pada penjadwalan ujian.

1. Asumsi

Asumsi yang diperlukan untuk membangun model optimisasi penjadwalan ujian yaitu sebagai berikut:

a. Satu *timeslot* mata kuliah yang diujiankan sama dengan $2\frac{1}{2}$ jam atau 150 menit.
b. Slot waktu cukup untuk menjadwalkan seluruh mata kuliah yang diujiankan.

2. Himpunan dan Parameter

Himpunan dan parameter yang digunakan dalam pemodelan untuk permasalahan ujian dapat ditunjukkan sebagai berikut

I : Himpunan mata kuliah yang diujikan, dengan I = {1, 2, 3, …, n}

T : Himpunan slot waktu, dengan T = {1, 2, 3, …, k}

H : Himpunan hari, dengan H = {1, 2, 3, …, h}

S : Himpunan mahasiswa yang mengikuti ujian, dengan S = {1, 2, …m}

M : Jumlah total mahasiswa

N : Jumlah ujian

K : Jumlah *timeslot*

R : Kapasistas total ruangan dalam 1 sesi

C_{ij} : Banyaknya mahasiswa yang mengikuti ujian matakuliah i dan j

V_{ij} : Parameter yang menunjukkan apakah mata kuliah yang diujiankan i dan j dijadwalkan pada *timeslot* yang sama

S_i : Banyaknya mahasiswa peserta ujian untuk mata kuliah ke-i

3. Variabel Keputusan

Variabel keputusan dari permasalahan penjadwalan ujian digunakan untuk memenuhi batasan yaitu memastikan bahwa setiap mata kuliah yang diujiankan terjadwal.

$$X_{it} = \begin{cases} 1, \textit{jika ujian i dijadwalkan pada timeslot t} \\ 0, \textit{yang lainnya} \end{cases}$$

4. Fungsi Tujuan

Terwujudnya jadwal ujian yang memadai dilakukan melalui optimisasi penjadwalan ujian dengan memperhatikan berbagai faktor yang mempengaruhinya. Hal tersebut diharapkan dapat berkorelasi positif dengan hasil capaian mahasiswa dalam ujian. Jeda antar ujian menjadi pertimbangan penting agar mahasiswa memiliki waktu cukup untuk belajar atau istirahat sebelum ujian selanjutnya dilaksanakan. Jumlah ujian yang diikuti seorang mahasiswa dalam sehari juga menjadi pertimbangan penting. Semakin banyak ujian yang diikuti maka semakin banyak pula materi yang harus ditinjau oleh mahasiswa. Oleh karena itu, fungsi tujuan dari permasalahan penjadwalan ujian

adalah meminimumkan pelanggaran *soft constraint* atau nilai *proximity* (P). Fungsi tujuan tersebut dinyatakan sebagai berikut:

$$\textbf{Meminimumkan}: \ P = P_1 + P_2$$

dengan,

P_1 = nilai penalti mahasiswa yang mengambil dua matakuliah berturut-turut dalam sehari

P_2 = nilai penalti mahasiswa yang mengambil dua matakuliah dalam sehari

Penalti yang pertama adalah menghitung jumlah mahasiswa yang dijadwalkan dua ujian berturut-turut dalam sehari yang dinyatakan dengan P_1. Persamaan matematis P_1 dinyatakan sebagai berikut:

$$P_1 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{s=1}^{M}\sum_{h=1}^{H} W_{sh}$$

dengan,

$s = \{1, 2, 3, \dots, M\}$ adalah urutan mahasiswa.

$h = \{1, 2, 3, \dots, H\}$ adalah urutan hari.

W_{sh} adalah nilai mahasiswa ke s pada hari ke h dalam variabel biner yang didefinisikan sebagai:

$$W_{sh} = \begin{cases} 1, jika\ pada\ hari\ h, mahasiswa\ s\ mengikuti\ 2\ ujian\ berturut \\ 0, jika\ tidak \end{cases}$$

Sedangkan P_2 digunakan untuk menghitung jumlah mahasiswa yang dijadwalkan lebih dari dua ujian dalam sehari. Persamaan matematis P_2 dinyatakan sebagai berikut:

$$P_2 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{s=1}^{M}\sum_{h=1}^{H} Y_{sh}$$

dengan,

$s = \{1, 2, 3, \dots, M\}$ adalah urutan mahasiswa

$h = \{1, 2, 3, \dots, H\}$ adalah urutan hari

Y_{sh} adalah nilai mahasiswa ke s pada hari ke h dalam variabel biner yang didefinisikan pada persamaan sebagai berikut:

$$Y_{sh} = \begin{cases} 1, & jika\ pada\ hari\ h, mahasiswa\ s\ mengikuti\ lebih\ dari\ dua\ ujian \\ 0, & jika\ tidak \end{cases}$$

5. Kendala (*Constrains*)

   Jadwal ujian yang dibuat diharuskan untuk memenuhi *hard constraints* sebagai berikut:

   a. Seluruh matakuliah yang diujiankan terjadwalkan sesuai dengan *timeslot* yang tersedia.

$$\sum_{t=1}^{k} X_{it} = 1 \quad ;\forall\ i\ \in I$$

   dengan,

   X_{it} : mata kuliah yang diujiankan i yang dijadwalkan pada *timeslot* t

   b. Tidak ada jadwal yang bentrok antara dua mata kuliah yang diujiankan yang diambil oleh mahasiswa yang sama.

$$\sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} C_{ij}V_{ij} = 0$$

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

   dengan,

   C_{ij} : Banyaknya mahasiswa yang mengikuti ujian mata kuliah i dan j

   V_{ij} : Parameter yang menunjukkan apakah mata kuliah yang diujiankan yang i dan j yang diambil setiap mahasiswa dijadwalkan pada *timeslot* yang sama.

   t_i : *timeslot* dimana mata kuliah yang diujiankan i dijadwalkan

   Persamaan diatas menunjukkan bahwa *timeslot* untuk mata kuliah yang diujiankan yang saling bentrok tidak boleh dijadwalkan bersamaan.

   c. Total mahasiswa peserta ujian tidak melebihi kapasitas total dalam satu sesi.

$$\sum_{i=1}^{n} S_i X_{it} \leq R \quad ;\forall\ t\ \in T$$

   dengan,

   S_i : banyak mahasiswa yang mengikuti mata kuliah yang diujiankan i

Selengkapnya, model optimisasi masalah penjadwalan ujian adalah sebagai berikut :

Meminimumkan : $P = P_1 + P_2$

dengan,

$$P_1 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{s=1}^{M}\sum_{h=1}^{H} W_{sh}$$

$$P_2 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{s=1}^{M}\sum_{h=1}^{H} Y_{sh}$$

Terhadap :

$$\sum_{t=1}^{k} X_{it} = 1 \quad ;\forall\ i\ \in I$$

$$\sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} C_{ij}V_{ij} = 0$$

$$\sum_{i=1}^{n} S_i X_{it} \leq R \quad ;\forall\ t\ \in T$$

3.4. Teknik Penyelesaian Model

Terdapat berberapa metode yang dapat digunakan untuk menyelesaikan masalah penjadwalan ujian, diantaranya adalah metode *heuristic*, metode *meta-heuristic* hingga metode *hyper-heuristic*. *Graph Coloring* merupakan metode *heuristic* sederhana dengan mendefinisikan masalah penjadwalan yang dibahas dengan menggunakan graf dan mendefinisikan batasan melalui hubungan antar simpul. Simpul dalam graf merepresentasikan ujian, sisi merepresentasikan setiap ujian yang memiliki konflik, dan warna merepresentasikan *time*slot. Untuk menjadwalkan ujian, setiap ujian yang saling bentrok akan dihubungkan dengan sebuah sisi. Kemudian setiap simpul akan diwarnai atau akan diberi *timeslot* yang berbeda dengan simpul yang bertetangga dengannya. Metode ini sudah dipastikan tidak akan melanggar *hard constraints*. Kendala dalam penjadwalan mata kuliah dengan metode ini adalah masih ada kecendrungan terjadinya pelanggaran *soft constraints*, misal kurangnya slot waktu ujian, sehingga seharusnya mahasiswa

tidak boleh mengikuti dua ujian berturut-turut, namun tetap dijadwalkan ujian berturut-turut dengan melanggar *soft constraints*.

Great Deluge merupakan algoritma *meta-heuristic local search* yang dikembangkan oleh Dueck (Mandal & Kahar, 2015). Algoritma ini telah diusulkan untuk meningkatkan kualitas solusi awal. Analogi algoritma *Great Deluge* diambil dari seorang pendaki bukit yang mencari tempat yang lebih tinggi untuk menghindari permukaan air saat terjadinya banjir (Mandal A.K, 2020). Algoritma *Great Deluge* berusaha memberikan pencarian solusi yang lebih luas (Mccollum, Mcmullan, Parkes, Burke, & Abdullah, 2009) dengan menerima solusi yang lebih buruk dari solusi sebelumnya, jika kualitas solusinya kurang dari atau sama dengan batas yang diberikan. Sehingga kelebihan algoritma *Great Deluge* adalah dapat menjelajahi seluruh domain, dan algoritma *Great Deluge* mampu menghindari jebakan optimum lokal.

Dalam penelitian ini, terdapat dua tahap untuk menjadwalkan ujian. Pertama, metode pewarnaan graf akan dipakai untuk menghasilkan solusi awal. Selanjutnya, solusi awal ini akan diperbaiki menggunakan Algoritma *Great Deluge Hyper-Heuristic*. Berikut adalah tahapan penjadwalan ujian.

1. Pembentukan Solusi Awal dengan Algoritma *Graph Colouring*

Pada algoritma *Graph Colouring* untuk memecahkan masalah penjadwalan ujian memiliki langkah awal yaitu merepresentasikan masalah dalam bentuk graf. Simpul merepresentasikan ujian, sisi merepresentasikan setiap ujian yang memiliki konflik, dan warna merepresentasikan *timeslot*. Untuk menjadwalkan ujian, setiap ujian yang saling bentrok akan dihubungkan dengan sebuah sisi. Kemudian setiap simpul akan diwarnai atau akan diberi *timeslot* yang berbeda dengan simpul yang bertetangga dengannya.

Algoritma *Graph Colouring* digunakan untuk mencari solusi awal yang nantinya akan menjadi masukan untuk optimasi algoritma selanjutnya. Peneliti menggunakan metode *Largest Enrollment* sebagai algoritma pewarnaan graf, yaitu dengan menjadwalkan terlebih dahulu mata kuliah yang diujiankan yang diambil oleh mahasiswa terbanyak. Algoritma *Largest Enrollment* (LE) merupakan algoritma yang prinsipnya berdasarkan pada banyak mahasiswa yang mengikuti suatu ujian. Pada algoritma ini ujian yang memiliki pendaftar ujian (mahasiswa)

paling banyak akan dijadwalkan terlebih dahulu. Langkah-langkah algoritma LE dapat dirumuskan sebagai berikut:

a. Urutkan himpunan ujian dari yang memiliki mahasiswa terbanyak.
b. Pilih ujian dengan mahasiswa terbanyak. Masukkan ujian tersebut ke *timeslot* 1. Masukkan ke dalam *timeslot* 1 ini, semua ujian yang mungkin sesuai dengan urutan mahasiswa terbanyak sehingga dua ujian yang bentrok mempunyai warna yang berbeda, kapasistas ruang dalam *timeslot* 1 cukup. Jika semua ujian sudah dijadwalkan, langkah selesai. Jika masih ada yang belum terjadwalkan, masukkan ke *timeslot* selanjutnya.
c. Urutkan kembali himpunan ujian yang belum dijadwalkan dari yang memiliki mahasiswa terbanyak. Pilih ujian dengan mahasiswa terbanyak selanjutnya, masukkan ke *timeslot* 2. Masukkan ke dalam *timeslot* 2 ini, semua ujian yang mungkin, sesuai dengan urutan mahasiswa terbanyak sehingga dua ujian yang bentrok mempunyai warna yang berbeda, dan kapasistas ruang dalam *timeslot* 2 cukup. Jika semua ujian sudah dijadwalkan, langkah selesai. Jika masih ada yang belum terjadwalkan, masukkan ke *timeslot* selanjutnya.
d. Ulangi langkah c sampai semua ujian dijadwalkan

2. Perbaikan Solusi dengan Algoritma *Great Deluge Hyper-Heuristic*

Solusi yang dihasilkan dari Langkah 1, selanjutkan akan diperbaiki dengan menggunakan Algoritma *Great Deluge Hyper-Heuristic*. Berikut adalah tahapan Algoritma *Great Deluge Hyper-Heuristic*.

1. Inisialisasi Variabel

Sebelum dapat menjalankan algoritma *Great Deluge Hyper-Heuristic*, beberapa variabel yang perlu diinisialisasi sebagai berikut

a. Nilai *Proximity* dari Solusi Awal (fs)
b. Nilai *Proximity* yang diinginkan (D)

   Nilai D dalam penelitian ini merupakan nilai kelipatan dari banyak mata kuliah dalam rentang 0 – fs.
c. Jumlah Iterasi (I)
d. Level (B)

   Level diinisiasikan sama dengan nilai *proximity* solusi awal.

$$B = fs$$

e. Pengurangan Level (ΔB)

ΔB dapat dihitung dengan pengurangan antara nilai *proximity* solusi awal dengan nilai *proximity* yang diinginkan dibagi dengan jumlah iterasi.

$$\Delta B = \frac{fs - D}{I}$$

2. Pemilihan Metode *Low Level Heuristic*

Jadwal yang telah dihasilkan algoritma *Graph Colouring* akan diubah untuk menemukan jadwal yang lebih baik dengan menggunakan pendekatan *low level heuristic*. Pendekatan *low lovel heuristic* yang digunakan penulis adalah:

L1: *Random Move*, yaitu memilih sebuah ujian lalu memindahkan ujian secara *random* dari *timeslot* asal ke *timeslot* lainnya.

L2: *Random Swap*, yaitu memilih dua ujian secara *random*, kemudian tukarkan kedua *timeslot* kedua ujian tersebut.

Pemilihan *low level heuristic* akan dipilih menggunakan *Simple Random* (SR). Langkah awalnya yaitu akan dibangkitkan secara acak sebuah bilangan bulat dalam rentang 0.0 - 1.0, kemudian akan dipilih *low level heuristic* sesuai dengan kriteria pemilihan sebagai berikut:

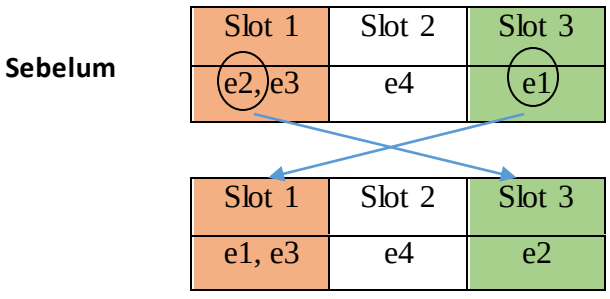
- Rentang [0, 0.5) maka *low level heuristic* yang dipakai adalah L1.
- Rentang [0.5, 1] maka *low level heuristic* yang dipakai adalah L2.

Low level heuristic yang terpilih akan digunakan dalam iterasi tersebut.

Gambar 3.1 merupakan contoh solusi awal yang diubah menggunakan pendekatan *low level heuristic* (*Random Swap*).

Contoh 3.1:

Misalkan terdapat 4 mata kuliah yang diujiankan, e1, e2, … e4. Mata kuliah e1, dan e2 bentrok dengan e4. Dari jadwal yang ada (solusi awal), akan dipilih secara acak 2 mata kuliah yang diujiankan.

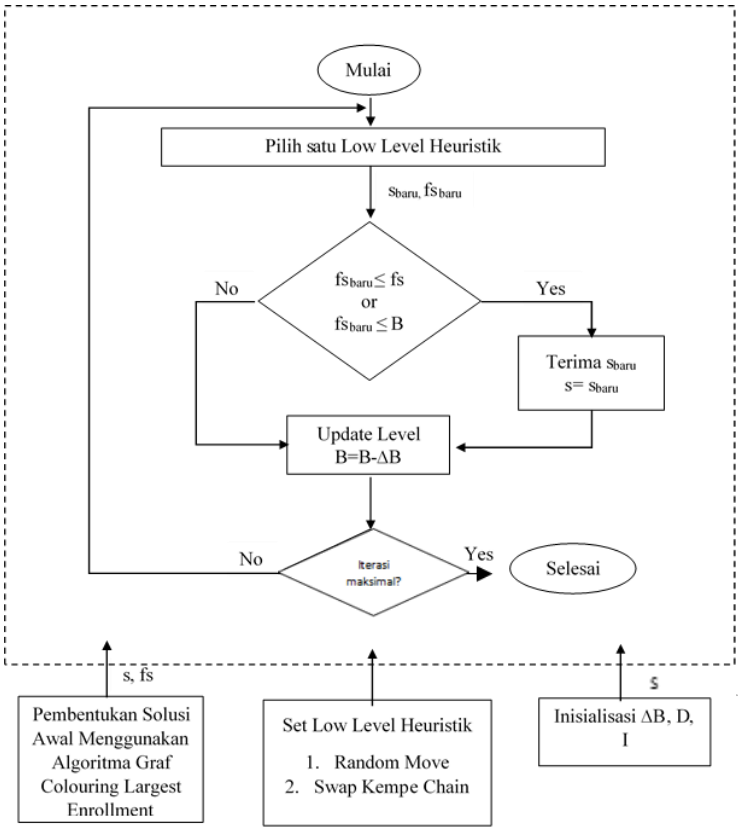


Gambar 3.1 Contoh *Random Swap*

Misalkan e1, dan e2 adalah mata kuliah yang terpilih. Pada solusi awal, mata kuliah e2 berada pada slot 1 dan mata kuliah e1 pada slot 3. Dengan menggunakan metode *Random Swap*, slot mata kuliah e2 akan ditukar dengan slot mata kuliah e1. Sehingga, mata kuliah e2 berada pada slot 3 dan mata kuliah e1 pada slot 1.

3. Penerimaan Solusi

Proses setelah memilih *low level heuristic* adalah untuk menentukan apakah solusi yang ada akan diterima atau tidak. Pada setiap iterasi, kandidat jadwal baru akan dibandingkan dengan nilai *proximity* jadwal sebelumnya. Apabila nilai *proximity* kandidat jadwal baru kurang dari nilai *proximity* jadwal terbaik sebelumnya atau kurang dari level, maka jadwal tersebut diterima. Selanjutnya level akan diturunkan oleh ΔB. Solusi optimal akan diperoleh dari iterasi terakhir algoritma *Great Deluge Hyper-Heuristic*. Berdasarkan penjelasan tersebut, maka cara kerja metode *Great Deluge Hyper-Heuristic* dapat digambarkan dalam *flowchart* pada Gambar 3.2.



Gambar 3.2 *Flowchart Great Deluge Hyper-Heuristic*

Untuk memperjelas langkah algoritma *Great Deluge Hyper-Heuristic*, maka akan diberikan contoh penyelesaian penjadwalan ujian 16 mahasiswa dan 10 mata kuliah yang diujiankan, seperti yang tercantum pada Tabel 3.1.

Tabel 3.1 Tabel Mata Kuliah yang Diujiankan yang Dikontrak Mahasiswa

Mahasiswa		Mata Kuliah yang Diujiankan									
		e1	e2	e3	e4	e5	e6	e7	e8	e9	e10
	1	1	1				1				1
	2	1				1	1		1		
	3	1				1			1		1
	4				1	1		1		1	
	5		1		1				1		
	6	1		1					1		
	7					1	1	1			1
	8	1	1								1
	9					1			1		1
	10	1					1				1
	11		1		1					1	
	12	1				1					1
	13	1	1						1	1	
	14	1		1							1
	15					1					1
	16			1							1
Jumlah Mahasiswa		9	5	3	3	7	4	2	6	3	10

Pada Tabel 3.1, menunjukkan mata kuliah yang diujiankan yang diambil oleh setiap mahasiswa. Perpotongan antara nomor mahasiswa dengan mata kuliah yang diujiankan (e) menunjukkan bahwa mahasiswa tersebut mengambil mata kuliah tersebut atau tidak. Angka 1 menunjukkan bahwa mahasiswa Y mengambil mata kuliah yang diujiankan X. Misalkan ujian akan diadakan selama 2 hari. Dimana dalam 1 hari terdapat 3 *timeslot* ujian. Maka terdapat 6 *timeslot* yaitu t1, t2, ..., t6.

Dalam 1 *timeslot*, kapasitas total ruang yang dapat dipakai adalah sebanyak 20 mahasiswa.

Berdasarkan Tabel 3.1, mata kuliah yang diujiankan yang tidak boleh bentrok dapat dituliskan pada Tabel 3.2.

Tabel 3.2 Mata Kuliah yang Diujiankan yang Tidak Boleh Bentrok

Mata Kuliah yang Diujiankan	Ujian yang Bentrok
e1	e2, e3, e5, e6, e8, e9, e10
e2	e1, e4, e6, e8, e9, e10
e3	e1, e8, e10
e4	e2, e5, e7, e8, e9
e5	e1, e4, e6, e7, e8, e9, e10
e6	e1, e2, e5, e7, e8, e10
e7	e4, e5, e6, e9, e10
e8	e1, e2, e3, e4, e5, e6, e9, e10
e9	e1, e2, e4, e5, e7, e8
e10	e1, e2, e3, e5, e6, e7, e8

Berikut adalah langkah untuk menjadwalkan ujian:

1. Pembentukan Solusi Awal dengan Algoritma *Graph Colouring*

Masalah penjadwalan ujian dapat direpresentasikan ke *Graph Colouring*, dimana simpul mewakili ujian, warna mewakili *timeslot* dan sisi mewakili konflik antara ujian satu dengan ujian yang lainnya. Didalam penelitian ini, penulis menggunakan metode *heuristic Largest Enrollment*, yaitu dengan menjadwalkan terlebih dahulu mata kuliah yang diujiankan yang diambil oleh mahasiswa terbanyak.

a. Iterasi 1

Dari Tabel 3.1, mata kuliah yang diujiankan, diurutkan dari yang memiliki mahasiswa terbanyak.

Tabel 3.3 Pengurutan Banyak Mahasiswa Tiap Mata kuliah Iterasi 1

Mata Kuliah yang Diujiankan	**Banyak Mahasiswa**
e10	10
e1	9
e5	7
e8	6
e2	5
e6	4
e3	3
e4	3
e9	3
e7	2

Karena e10 merupakan mata kuliah yang diujiankan yang berisi mahasiswa terbanyak. Maka e10 dijadwalkan terlebih dahulu. Masukkan e10 ke dalam *timeslot* t1. Berdasarkan tabel mata kuliah yang tidak boleh bentrok, mata kuliah e4 dan e9 merupakan ujian yang belum dijadwalkan ujian yang tidak bentrok dengan e10. Dari keduanya dipilih yang berisi mahasiswa terbanyak sesuai urutan Tabel 3.3. Maka terpilih e4. Karena kapasitas *timeslot* t1 masih cukup, masukkan e4 ke *timeslot* t1 dan kurangi kapasitas ruang *timeslot* t1 dengan banyak mahasiswa e10 dan e4. Karena tidak ada irisan mata kuliah yang tidak bentrok antara e10 dan e4, dan masih ada mata kuliah yang belum dijadwalkan ujian, maka lanjut ke iterasi 2.

Tabel 3.4 Penjadwalan *Graph Colouring* Iterasi 1

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata Kuliah yang Diujiankan	e10, e4					
Sisa Kapasitas Ruang	7	20	20	20	20	20

b. Iterasi 2

Perbaharui himpunan mata kuliah yang diujiankan, yang belum dijadwalkan. Urutkan kembali himpunan mata kuliah yang diujiankan, dari yang memiliki mahasiswa tebanyak.

Tabel 3.5 Pengurutan Banyak Mahasiswa Tiap Mata kuliah Iterasi 2

Mata Kuliah yang Diujiankan	**Banyak Mahasiswa**
e1	9
e5	7
e8	6
e2	5
e6	4
e3	3
e9	3
e7	2

Karena e1 merupakan mata kuliah yang diujiankan yang berisi mahasiswa terbanyak selanjutnya. Maka e1 dijadwalkan ke dalam *timeslot* t2. Berdasarkan tabel mata kuliah yang tidak boleh bentrok, hanya e7 yang merupakan mata kuliah yang belum dijadwalkan ujian yang tidak bentrok dengan e1. Karena kapasitas *timeslot* t2 masih cukup, masukkan juga e7 ke *timeslot* t2 dan kurangi kapasitas ruang *timeslot* t2 dengan banyak mahasiswa e1, dan e7. Karena masih ada mata kuliah yang belum dijadwalkan ujian, maka lanjut ke iterasi 3.

Tabel 3.6 Penjadwalan *Graph Colouring* Iterasi 2

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata Kuliah yang Diujiankan	e10,e4	e1,e7				
Sisa Kapasitas Ruang	7	9	20	20	20	20

c. Iterasi 3

Perbaharui himpunan mata kuliah yang diujiankan, yang belum dijadwalkan. Urutkan kembali himpunan mata kuliah yang diujiankan, dari yang memiliki mahasiswa tebanyak.

Tabel 3.7 Pengurutan Banyak Mahasiswa Tiap Mata kuliah Iterasi 3

Mata Kuliah yang Diujiankan	**Banyak Mahasiswa**
e5	7
e8	6
e2	5
e6	4
e3	3
e9	3

Karena e5 merupakan mata kuliah yang diujiankan yang berisi mahasiswa terbanyak selanjutnya. Maka e5 dijadwalkan ke dalam *timeslot* t3. Berdasarkan tabel mata kuliah yang tidak boleh bentrok, mata kuliah e2 dan e3 merupakan ujian yang belum dijadwalkan ujian yang tidak bentrok dengan e5. Dari keduanya dipilih yang berisi mahasiswa terbanyak sesuai urutan tabel 3.7. Maka terpilih e2. Karena kapasitas *timeslot* t3 masih cukup, masukkan e2 ke *timeslot* t3. e3 merupakan mata kuliah yang tidak bentrok dengan e5 dan e2, dan karena kapasitas ruangan masih cukup, maka masukkan juga e3 ke *timeslot* t3. Kemudian kurangi kapasitas ruang *timeslot* t3 dengan banyak mahasiswa e5, e2, dan e3. Karena tidak ada mata kuliah yang tidak bentrok antara e5, e2 dan e3, dan masih ada mata kuliah yang belum dijadwalkan ujian, maka lanjut ke iterasi 4.

Tabel 3.8 Penjadwalan *Graph Colouring* Iterasi 3

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e10,e4	e1,e7	e5, e2, e3			
Sisa Kapasitas Ruang	7	9	5	20	20	20

d. Iterasi 4

Perbaharui himpunan mata kuliah yang diujiankan, yang belum dijadwalkan. Urutkan kembali himpunan mata kuliah yang diujiankan, dari yang memiliki mahasiswa tebanyak.

Tabel 3.9 Pengurutan Banyak Mahasiswa Tiap Mata kuliah Iterasi 4

Mata Kuliah yang Diujiankan	**Banyak Mahasiswa**
e8	6
e6	4
e9	3

Karena e8 merupakan mata kuliah yang diujiankan yang berisi mahasiswa terbanyak selanjutnya. Maka masukkan e8 ke *timeslot* t4 dan kurangi kapasitas *timeslot* t4 dengan banyak mahasiswa e8. Karena tidak ada mata kuliah yang tidak bentrok antara e8, dan masih ada mata kuliah yang belum dijadwalkan ujian, maka lanjut ke iterasi 5.

Tabel 3.10 Penjadwalan *Graph Colouring* Iterasi 4

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e10,e4	e1,e7	e5, e2, e3	e8		
Sisa Kapasitas Ruang	7	9	5	14	20	20

e. Iterasi 5

Perbaharui himpunan mata kuliah yang diujiankan, yang belum dijadwalkan. Urutkan kembali himpunan mata kuliah yang diujiankan, dari yang memiliki mahasiswa tebanyak.

Tabel 3.11 Pengurutan Banyak Mahasiswa Tiap Mata kuliah Iterasi 5

Mata Kuliah yang Diujiankan	**Banyak Mahasiswa**
e6	4
e3	3

Karena e6 merupakan mata kuliah yang diujiankan yang berisi mahasiswa terbanyak selanjutnya. Maka e6 dijadwalkan ke dalam *timeslot* t5. Berdasarkan

tabel mata kuliah yang tidak boleh bentrok, e9 merupakan mata kuliah yang belum terjadwalkan yang tidak bentrok dengan e6. Karena kapasitas *timeslot* t5 masih cukup, masukkan juga e9 ke *timeslot* t5 dan kurangi kapasitas ruang *timeslot* t5 dengan banyak mahasiswa e6, dan e9.

Tabel 3.12 Penjadwalan *Graph Colouring* Iterasi 5

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e10,e4	e1,e7	e5, e2, e3	e8	e6, e9	
Sisa Kapasitas Ruang	7	9	5	14	13	20

Karena tidak ada mata kuliah yang belum dijadwalkan ujian, maka pengerjaan berhenti. Didapat hasil dari pengerjaan adalah sebagai berikut:

Tabel 3.13 Hasil Penjadwalan *Graph Colouring*

Timeslot	**Mata Kuliah yang Diujiankan**
t1	e10
	e4
t2	e1
	e7
t3	e5
	e2
	e3
t4	e8
t5	e6
	e9
t6	

Dari tabel diatas, dapat dilihat bahwa jadwal tidak melanggar *hard constraint* sehingga jadwal dapat dikatakan sudah optimal. Untuk mencari nilai *proximity*, harus diketahui *timeslot* berapa saja yang diambil oleh setiap mahasiswa.

Tabel 3.14 *Timeslot* Setiap Mahasiswa

Mahasiswa	**Mata Kuliah yang Diujiankan**	***Timeslot***
1	e1, e2, e6, e10	t2, t3, t5, t1
2	e1, e5, e6, e8	t2, t3, t5, t4
3	e1, e5, e8, e10	t2, t3, t4, t1
4	e4, e5, e7, e9	t1, t3, t2, t5
5	e2, e4, e8	t3, t1, t4
6	e1, e3, e8	t2, t3, t4
7	e5, e6, e7, e10	t3, t5, t2, t1
8	e1, e2, e10	t2, t3, t1
9	e5, e8, e10	t3, t4, t1
10	e1, e6, e10	t2, t5, t1
11	e2, e4, e9	t3, t1, t5
12	e1, e5, e10	t2, t3, t1
13	e1, e2, e8, e9	t2, t3, t4, t5
14	e1, e3, e10	t2, t3, t1
15	e5, e10	t3, t1
16	e3, e10	t3, t1

Diketahui bahwa *timeslot* t1, t2, t3 berada pada hari ke-1, dan *timeslot* t4, t5, t6 berada pada hari ke-2. Maka untuk menghitung nilai *proximity* terdapat 2 tahap, yaitu sebagai berikut:

1. Menghitung jumlah mahasiswa yang dijadwalkan 2 ujian berturut turut dalam sehari (P_1).

Setiap mahasiswa yang memiliki 2 ujian berturut-turut dalam sehari, maka diberi nilai 1. Sebaliknya, diberi nilai 0.

Tabel 3.15 Nilai Harian P1 Jadwal 1

Mahasiswa	***Timeslot* Hari 1**	**Nilai Hari 1**	***Timeslot* Hari 2**	**Nilai Hari 2**
1	t1, t2, t3	1	t5	0
2	t2,t3	1	t4, t5	1
3	t1, t2, t3	1	t4	0
4	t1, t2, t3	1	t4	0
5	t1, t3	0	t4	0
6	t2, t3	1	t4	0
7	t1, t2, t3	1	t5	0
8	t1, t2, t3	1	-	0
9	t1, t3	0	t4	0
10	t1, t2	1	t5	0
11	t1, t3	0	t5	0
12	t1, t2, t3	1	-	0
13	t2, t3	1	t4, t5	1
14	t1, t2, t3	1	-	0
15	t1, t3	0	-	0
16	t1, t3	0	-	0

Sehingga diperoleh nilai P_1 yaitu:

$$P_1 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{l=1}^{M}\sum_{h=1}^{H} W_{sh}$$

$$= 10 \cdot ((1+0)+(1+1)+(1+0)+(1+0)+(0+0)+(1+0) + (1+0)+(1+0)+(0+0)+(1+0)+(0+0) + (1+0)+(1+1)+(1+0)+(0+0)+(0+0))$$

$$P_1 = 130$$

2. Menghitung jumlah mahasiswa yang dijadwalkan 2 ujian dalam satu hari (P_2). Setiap mahasiswa yang memiliki lebih dari 2 ujian dalam sehari, maka diberi nilai 1. Sebaliknya, diberi nilai 0.

Tabel 3.16 Nilai Harian P2 Jadwal 1

Mahasiswa	Jumlah Ujian Hari 1	Nilai Hari 1	Jumlah Ujian Hari 2	Nilai Hari 2
1	3	1	1	0
2	2	0	2	0
3	3	1	1	0
4	3	1	1	0
5	2	0	1	0
6	2	0	1	0
7	3	1	1	0
8	3	1	0	0
9	2	0	1	0
10	2	0	1	0
11	2	0	1	0
12	3	1	0	0
13	2	0	2	0
14	3	1	0	0
15	2	0	0	0
16	2	0	0	0

Sehingga diperoleh nilai P_2 yaitu:

$$P_2 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{l=1}^{M}\sum_{h=1}^{H} Y_{sh}$$

$$= 10 \cdot ((1+0)+(0+0)+(1+0)+(1+0)+(0+0)+(0+0)$$
$$+(0+0)+(1+0)+(0+0)+(0+0)+(1+0)$$
$$+(1+0)+(1+0)+(0+0)+(0+0)+(0+0))$$

$$= 70$$

Maka,

$$P = P_1 + P_2$$

$$= 130 + 70 = 200$$

Didapat bahwa nilai *proximity* (P) adalah 200

2. Perbaikan Solusi

Solusi yang dihasilkan dari langkah 1, selanjutkan akan diperbaiki dengan menggunakan Algoritma *Great Deluge Hyper-Heuristic*. Berikut adalah tahapan Algoritma *Great Deluge Hyper-Heuristic.*

1. Inisialisasi Variabel

Berdasarkan hasil dari langkah 1, didapatkan

Nilai *proximity* solusi awal (Fs) = 200

Misalkan

- Nilai *proximity* yang diinginkan (D) dalam contoh ini merupakan kelipatan 10 yang berada dalam rentang [0, 200], dengan D = 20
- Iterasi (I)= 2
- Level diinisiasikan sama dengan nilai *proximity* solusi awal

  B = fs = 200
- ΔB = (fs-D)/I

  = (200-20)/2 = 90

a. Iterasi 1

1. Pemilihan *low level heuristic*

Misalkan,

L1 = *Random Move*

L2 = *Random Swap*

Cara pemilihan *low level heuristic* adalah akan dibangkitkan secara acak sebuah bilangan bulat dalam rentang 0.0 - 1.0 dimana kriteria pemilihan sebagai berikut:

- Rentang [0, 0.5) maka *low level heuristic* yang dipakai adalah L1.
- Rentang [0.5, 1] maka *low level heuristic* yang dipakai adalah L2.

Misalkan angka yang dibangkitkan saat iterasi 1 adalah 0.2, maka *low level heuristic* yang dipakai adalah L1, yaitu *Random Move*.

Misalkan ujian yang terpilih adalah e10, dan *timeslot* yang terpilih adalah 6. Maka e10 akan dipindahkan ke *timeslot 6* jika tidak bentrok dan masih memenuhi kapasitas ruang.

Jadwal sebelum dan setelah memakai *low level heuristic* iterasi 1 dapat dilihat pada Tabel 3.17 dan Tabel 3.18.

Tabel 3.17 Jadwal Sebelum Menggunakan *Low Level Heuristic* Iterasi 1

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e10,e4	e1,e7	e5, e2, e3	e8	e6, e9	
Sisa Kapasitas Ruang	7	9	5	14	13	20

Tabel 3.18 Jadwal Setelah Menggunakan *Low Level Heuristic* Iterasi 1

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e4	e1,e7	e5, e2, e3	e8	e6, e9	e10
Sisa Kapasitas Ruang	17	9	5	14	13	10

Dari hasil *random move*, *timeslot* ujian tiap mahasiswa akan berubah yang dapat dilihat sebagai berikut:

Tabel 3.19 Timeslot Mahasiswa Setelah Menggunakan *Low Level Heuristic* Iterasi 1

Mahasiswa	**Mata Kuliah yang Diujiankan**	**Timeslot**
1	e1, e2, e6, e10	t2, t3, t5, t6
2	e1, e5, e6, e8	t2, t3, t5, t4
3	e1, e5, e8, e10	t2, t3, t4, t6
4	e4, e5, e7, e9	t1, t3, t2, t5
5	e2, e4, e8	t3, t1, t4
6	e1, e3, e8	t2, t3, t4
7	e5, e6, e7, e10	t3, t5, t2, t6
8	e1, e2, e10	t2, t3, t6
9	e5, e8, e10	t3, t4, t6
10	e1, e6, e10	t2, t5, t6
11	e2, e4, e9	t3, t1, t5
12	e1, e5, e10	t2, t3, t6
13	e1, e2, e8, e9	t2, t3, t4, t5
14	e1, e3, e10	t2, t3, t6
15	e5, e10	t3, t6
16	e3, e10	t3, t6

Sama seperti sebelumnya, untuk menghitung nilai *proximity* ada 2 tahap, yaitu

1. Menghitung jumlah mahasiswa yang dijadwalkan 2 ujian berturut turut dalam sehari (P_1).

Setiap mahasiswa yang memiliki 2 ujian berturut-turut dalam sehari, maka diberi nilai 1. Sebaliknya, diberi nilai 0.

Tabel 3.20 Nilai Harian P1 Jadwal 2

Mahasiswa	Timeslot Hari 1	Nilai Hari 1	Timeslot Hari 2	Nilai Hari 2
1	t2, t3	1	t5, t6	1
2	t2,t3	1	t4, t5	1
3	t2, t3	1	t4, t6	0
4	t1, t2, t3	1	t5	0
5	t1, t3	0	t4	0
6	t2, t3	1	t4	0
7	t2, t3	1	t5, t6	1
8	t2, t3	1	t6	0
9	t3	0	t4, t6	0
10	t2	0	t5, t6	1
11	t1, t3	0	t5	0
12	t2, t3	1	t6	0
13	t2, t3	1	t4, t5	1
14	t2, t3	1	t6	0
15	t3	0	t6	0
16	t3	0	t6	0

Sehingga diperoleh nilai P_1 yaitu:

$$P_1 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{l=1}^{M}\sum_{h=1}^{H} W_{sh}$$

$$= 10 \cdot ((1+1)+(1+1)+(1+0)+(1+0)+(0+0)+(1+0) + (1+1)+(1+0)+(0+0)+(0+1)+(0+0) + (1+0)+(1+1)+(1+0)+(0+0)+(0+0))$$

$$= 150$$

2. Menghitung jumlah mahasiswa yang dijadwalkan 2 ujian dalam satu hari (P_2).

Setiap mahasiswa yang memiliki lebih dari 2 ujian dalam sehari, maka diberi nilai 1. Sebaliknya, diberi nilai 0.

Tabel 3.21 Nilai Harian P2 Jadwal 2

Mahasiswa	Jumlah Ujian Hari 1	Nilai Hari 1	Jumlah Ujian Hari 2	Nilai Hari 2
1	2	0	2	0
2	2	0	2	0
3	2	0	2	0
4	3	1	1	0
5	2	0	1	0
6	2	0	1	0
7	2	0	2	0
8	2	0	1	0
9	1	0	2	0
10	1	0	2	0
11	2	0	1	0
12	2	0	1	0
13	2	0	2	0
14	2	0	1	0
15	1	0	1	0
16	0	0	2	0

Sehingga diperoleh nilai P_2 yaitu:

$$P_2 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{s=1}^{M}\sum_{h=1}^{H} Y_{sh}$$

$$= 10 \cdot ((0+0)+(0+0)+(0+0)+(1+0)+(0+0)+(0+0) + (0+0)+(0+0)+(0+0)+(0+0)+(0+0) + (0+0)+(0+0)+(0+0)+(0+0)+(0+0))$$

$$= 10$$

Maka,

$P = P_1 + P_2$

$= 150 + 10 = 160$

Maka nilai *proximity*nya (fs baru) adalah 160

2. Penerimaan Solusi

Proses setelah memilih *low level heuristic* adalah untuk menentukan apakah solusi yang ada akan diterima atau tidak. Kriteria pemilihan solusi memakai kriteria algoritma *Great Deluge* yaitu:

Jika fs baru < fs atau fs baru < B maka solusi diterima

Diketahui bahwa

Fs = 200

Fs baru = 160

Fs baru < fs atau 160 < 200, Benar. Maka solusi baru diterima.

Batas akan dikurangi oleh ΔB, sehingga

B = 200-90 = 110

b. Iterasi 2

1. Pemilihan *low level heuristic*

Misalkan angka yang dibangkitkan saat iterasi 2 adalah 0.9, maka *low level heuristic* yang dipakai adalah L1, yaitu *Random Swap*.

Misalkan terpilih e4 dan e9. Maka *timeslot* keduanya ditukar. e9 berada pada *timeslot* t1 sedangkan e4 pada *timeslot* t5.

Jadwal sebelum dan setelah memakai *low level heuristic* iterasi 2 dapat dilihat pada Tabel 3.22 dan Tabel 3.23.

Tabel 3.22 Jadwal Sebelum Menggunakan *Low Level Heuristic* Iterasi 2

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e4	e1,e7	e5, e2, e3	e8	e6, e9	e10
Sisa Kapasitas Ruang	17	9	5	14	13	10

Tabel 3.23 Jadwal Setelah Menggunakan *Low Level Heuristic* Iterasi 2

Timeslot	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**
Mata kuliah yang diujiankan	e9	e1,e7	e5, e2, e3	e8	e6, e4	e10
Sisa Kapasitas Ruang	17	9	5	14	13	10

Dari hasil *Random Swap*, terdapat perubahan *timeslot* ujian tiap mahasiswa yang dapat dilihat sebagai berikut

Tabel 3.24 Timeslot Mahasiswa Setelah Menggunakan *Low Level Heuristic* Iterasi 2

Mahasiswa	**Mata Kuliah yang Diujiankan**	**Timeslot**
1	e1, e2, e6, e10	t2, t3, t5, t6
2	e1, e5, e6, e8	t2, t3, t5, t4
3	e1, e5, e8, e10	t2, t3, t4, t6
4	e4, e5, e7, e9	t5, t3, t2, t1
5	e2, e4, e8	t3, t5, t4
6	e1, e3, e8	t2, t3, t4
7	e5, e6, e7, e10	t3, t5, t2, t6
8	e1, e2, e10	t2, t3, t6
9	e5, e8, e10	t3, t4, t6
10	e1, e6, e10	t2, t5, t6
11	e2, e4, e9	t3, t5, t1
12	e1, e5, e10	t2, t3, t6
13	e1, e2, e8, e9	t2, t3, t4, t1
14	e1, e3, e10	t2, t3, t6
15	e5, e10	t3, t6
16	e3, e10	t3, t6

Sama seperti sebelumnya, untuk menghitung nilai *proximity* ada 2 tahap, yaitu

1. Menghitung jumlah mahasiswa yang dijadwalkan 2 ujian berturut turut dalam sehari (P_1).

Setiap mahasiswa yang memiliki 2 ujian berturut-turut dalam sehari, maka diberi nilai 1. Sebaliknya, diberi nilai 0.

Tabel 3.25 Nilai Harian P1 Jadwal 3

Mahasiswa	Timeslot Hari 1	Nilai Hari 1	Timeslot Hari 2	Nilai Hari 2
1	t2, t3	1	t5, t6	1
2	t2,t3	1	t4, t5	1
3	t2, t3	1	t4, t6	0
4	t1, t2, t3	1	t5	0
5	t3	0	t4, t5	1
6	t2, t3	1	t4	0
7	t2, t3	1	t5, t6	1
8	t2, t3	1	t6	0
9	t3	0	t4, t6	0
10	t2	0	t5, t6	1
11	t1, t3	0	t5	0
12	t2, t3	1	t6	0
13	t1, t2, t3	1	t4	0
14	t2, t3	1	t6	0
15	t3	0	t6	0
16	t3	0	t6	0

Sehingga diperoleh nilai P_1 yaitu:

$$P_1 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{l=1}^{M}\sum_{h=1}^{H} W_{sh}$$

$$= 10 \cdot ((1+1)+(1+1)+(1+0)+(1+0)+(0+0)+(1+0) + (1+1)+(1+0)+(0+0)+(0+1)+(0+0) + (1+0)+(1+0)+(1+0)+(0+0)+(0+0))$$

$$= 150$$

2. Menghitung jumlah mahasiswa yang dijadwalkan 2 ujian dalam satu hari (P_2).

Setiap mahasiswa yang memiliki lebih dari 2 ujian dalam sehari, maka diberi nilai 1. Sebaliknya, diberi nilai 0.

Tabel 3.26 Nilai Harian P2 Jadwal 3

Mahasiswa	Jumlah Ujian Hari 1	Nilai Hari 1	Jumlah Ujian Hari 2	Nilai Hari 2
1	2	0	2	0
2	2	0	2	0
3	2	0	2	0
4	3	1	1	0
5	2	0	1	0
6	2	0	1	0
7	2	0	2	0
8	2	0	1	0
9	1	0	2	0
10	1	0	2	0
11	2	0	1	0
12	2	0	1	0
13	3	1	1	0
14	2	0	1	0
15	1	0	1	0
16	0	0	2	0

Sehingga diperoleh nilai P_2 yaitu:

$$P_2 = \sum_{i=1}^{n}\sum_{t=1}^{k} X_{it} \sum_{l=1}^{M}\sum_{h=1}^{H} Y_{sh}$$

$$= 10 \cdot ((0+0)+(0+0)+(0+0)+(1+0)+(0+0)+(0+0) + (0+0)+(0+0)+(0+0)+(0+0)+(0+0) + (0+0)+(1+0)+(0+0)+(0+0)+(0+0))$$

$$= 20$$

Maka,

$P = P_1 + P_2$

$= 150 + 20 = 170$

Maka nilai *proximity*nya (fs baru) adalah 170

2. Penerimaan Solusi

Proses setelah memilih *low level heuristic* adalah untuk menentukan apakah solusi yang ada akan diterima atau tidak. Kriteria pemilihan solusi memakai kriteria algoritma *Great Deluge* yaitu,

Jika fs baru < fs atau fs baru < B maka solusi diterima

Diketahui bahwa

Fs = 160

Fs baru = 170

Batas = 110

Fs baru < Fs atau 170 < 160, Salah. Maka solusi baru ditolak.

Karena iterasi yang diinginkan telah tercapai, yaitu 2 iterasi sehingga iterasi telah terpenuhi. Dari keseluruhan jadwal yang didapat, maka dipilih solusi yang memiliki *nilai proximity* terkecil yaitu solusi dari hasil iterasi 1, karena pada saat iterasi 2, solusi baru ditolak atau dengan kata lain, solusinya tidak lebih baik dari solusi sebelumnya. Solusi yang memiliki nilai *proximity* terbaik yaitu 160, dimana jadwal ujian dapat dilihat pada Tabel 3.18, atau dapat disimpulkan dalam tabel berikut.

Tabel 3.27 Hasil Penjadwalan *Great Deluge Hyper-Heuristic*

Timeslot	Mata Kuliah yang Diujiankan
t1	e4
t2	e1
	e7
t3	e5
	e2
	e3
t4	e8
t5	e6
	e9
t6	e10