BAB IV
PEMBAHASAN

Bab ini berisi data penelitian, tahap implementasi metode *Great Deluge Hyper-Heuristic* dalam penyelesaian penjadwalan ujian, validasi, dan hasil implementasi metode *Great Deluge Hyper-Heuristic* dalam penyelesaian penjadwalan ujian.

4.1. Data Penelitian

Data yang digunakan adalah data mahasiswa, dan data mata kuliah berasal dari salah satu lembaga pendidikan dalam dataset Carter (Toronto) untuk penjadwalan ujian yaitu data yor-83 atau data York Mills Collegiate Institute, North York, Kanada. Dataset tersebut dapat diakses pada *link* http://www.cs.nott.ac.uk/~pszrq/data.htm.

1. Data Mahasiswa

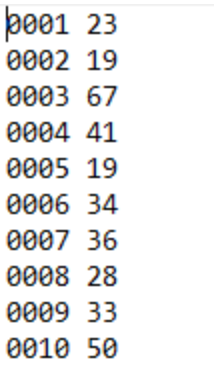
Data mahasiswa York Mills Collegiate Institute yang digunakan dalam penelitian ini terdiri dari 941 mahasiswa. Data tersebut berupa nomor mahasiswa dan mata kuliah yang diambil masing-masing mahasiswa. Gambar 4.1 merupakan potongan data yang telah di*download*. Data tersebut adalah data mata kuliah yang diikuti oleh setiap mahasiswa, dimana setiap nomor baris merepresentasikan urutan mahasiswa. Misal angka 0040 pada baris 1 memiliki arti bahwa mahasiswa 1 mengikuti 2 mata kuliah yang diujiankan yaitu mata kuliah 40 dan 145. Demikian pula pada baris 5 memiliki arti bahwa mahasiswa 5 mengikuti 3 mata kuliah yang diujiankan yaitu mata kuliah 40, 96, dan 116.

```
0040 0145
0009 0052 0102 0179
0033 0069 0083 0105 0114 0122 0134 0156
0012 0016 0018 0032 0122 0126 0134 0161
0037 0103 0106 0118 0123 0148 0158 0171
0040 0096 0116
0004 0008 0014 0017 0035 0139
0088 0101 0102 0145 0151
0037 0119 0141 0148 0171 0172 0177
0034 0064 0084 0095 0114 0122 0136 0155
```

Gambar 4.1 Contoh Data dari YOR-83 Carter (Toronto) 1

2. Data Mata Kuliah yang Diujikan

Data mata kuliah berisi semua mata kuliah yang diujiankan di York Mills Collegiate Institute serta banyak mahasiswa yang mengikuti mata kuliah tersebut. Gambar 4.2 merupakan contoh data yang telah didownload pada link yang telah disebutkan sebelumnya dari Carter yang berisi data mengenai banyaknya mahasiswa dari setiap mata kuliah yang diujiankan. Kolom 1 merepresentasikan mata kuliah yang telah dikodekan dengan format urutan numerik 4 digit dan kolom 2 merepresentasikan banyak mahasiswanya. Misalkan pada baris 1, angka 0001 pada kolom 1 adalah mata kuliah 1 dan angka 23 pada kolom 2 adalah banyak mahasiswa yang mengikuti mata kuliah 1.



Gambar 4.2 Contoh Potongan Data dari YOR-83 Carter (Toronto) 2

Tabel 4.1 menunjukan karakteristik Dataset Yor-83. Data yang diperoleh terdiri dari banyak mahasiswa, banyak mata kuliah yang diujiankan, dan banyak *timeslot*. Berdasarkan hasil pengujian, 21 *timeslot* yang tersedia menghasilkan solusi awal yang tidak layak (melanggar 1 buah *hard constraints*). Oleh karena itu perlu dilakukan analisis terhadap kebutuhan *timeslot* yang akan menghasilkan solusi awal yang layak. Langkah pertama adalah menetapkan kapasitas ruang yang digunakan pada penelitian ini yaitu 70% dari banyak mahasiswa sebagaimana juga dilakukan dalam penelitian Masri Ayob (2007). Peneliti mengvariasikan banyak *timeslot* mulai dari 21 timeslot hingga solusi layak ditemukan yaitu 23 timeslot. Percobaan banyak *timeslot* dapat dilihat pada Tabel 4.2.

Dalam penelitian Burke (1999), ujian dilaksanakan dari hari Senin - Sabtu dimana setiap harinya terdapat 3 *timeslot*. Berdasarkan penelitian tersebut, penulis menghitung jumlah hari yang digunakan untuk ujian.

Tabel 4.1 Informasi Karakteristik Dataset YOR-83 Carter

Banyak Mata Kuliah yang Diujiankan	181

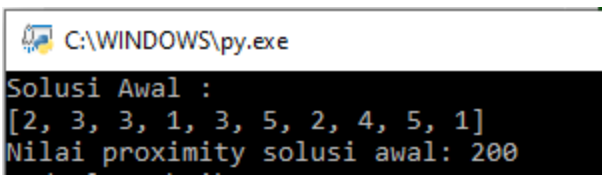
Banyak Siswa	941
Banyak *Timeslot*	23
Periode Ujian (Hari)	8
Kapasitas Ruang	659

Tabel 4.2 Solusi layak dari Dataset YOR-83 Carter

Timeslot	**Banyak mata kuliah yang belum terjadwal**	**Keterangan**
21	7	Solusi tidak layak
22	3	Solusi tidak layak
23	0	Solusi layak

4.2. 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 yang dihasilkan oleh *software* Python 3.10 dengan solusi hasil perhitungan manual dengan mengambil contoh kasus pada bab III. Berdasarkan hasil pengujian, diperoleh bahwa solusi awal yang dihasilkan metode *Graph Colouring* memakai program dengan pengerjaan manual yang dapat dilihat pada Tabel 3.13 adalah sama. Gambar 4.3 menunjukkan hasil *output* program penjadwalan contoh kasus pada Bab III. Pada gambar terdapat sebuah array yang berisi 10 angka. Angka pertama dalam array merupakan *timeslot* untuk mata kuliah 1, angka kedua merupakan *timeslot* untuk mata kuliah 2, dan seterusnya. Contoh, angka pertama dalam array adalah 2, maka mata kuliah 1 berada pada *timeslot* 2.



Gambar 4.3 Validasi Program

Kemudian validasi untuk algoritma *Great Deluge*, karena algoritma *Great Deluge Hyper-Heuristic* bersifat stokastik, penulis memanipulasi solusi awal pada penelitian sehingga mendapatkan jadwal yang sama dengan setelah memakai *low*

level heuristic iterasi 1 dan iterasi 2 seperti contoh kasus di Bab III. Gambar 4.4 menunjukkan hasil *output* program penjadwalan menggunakan *Great Deluge Hyper-Heuristic* untuk contoh kasus pada Bab III.

```
Jadwal solusi awal : [2, 3, 3, 1, 3, 5, 2, 4, 5, 1]
Nilai proximity solusi awal : 200
Jadwal setelah pakai low level heuristik iterasi 1 :
[2, 3, 3, 1, 3, 5, 2, 4, 5, 6]
Nilai proximity iterasi 1 : 160
Jadwal terbaik antara solusi awal dan hasil iterasi 1 adalah :
[2, 3, 3, 1, 3, 5, 2, 4, 5, 6]
Nilai proximity jadwal terbaik sementara: 160
Jadwal setelah pakai low level heuristik iterasi 2 :
[2, 3, 3, 5, 3, 5, 2, 4, 1, 6]
Nilai proximity iterasi 2 : 170
Jadwal terbaik dari 2 iterasi adalah :
[2, 3, 3, 5, 3, 5, 2, 4, 1, 6]
Nilai proximity jadwal terbaik : 160
```

Gambar 4.4 Hasil Penjadwalan *Great Deluge Hyper-Heuristic*

Dari gambar dapat dilihat bahwa untuk penjadwalan program setelah pemakaian *low level heuristic* iterasi 1 dan iterasi 2 memiliki nilai *proximity* yang sama dengan perhitungan manual. Kemudian, setelah memakai kriteria pemilihan solusi *Great Deluge*, terpilih jadwal terbaik yaitu hasil iterasi 1 dengan nilai *proximity* 160. Berdasarkan hasil pengujian, diperoleh bahwa solusi terbaik yang dihasilkan metode *Great Deluge* memakai program dengan pengerjaan manual yang dapat dilihat pada Tabel 3.27 adalah sama.

4.3. Tahapan Implementasi

Subbab ini membahas mengenai tahapan implementasi dari algoritma *Great Deluge Hyper-Heuristic* untuk masalah penjadwalan ujian di York Mills Collegiate Institute. Secara garis besar, tahapan Algoritma *Great Deluge Hyper-Heuristic* dalam penelitian ini mengikuti *flowchart* pada Gambar 3.2. Dalam penelitian ini, terdapat dua tahap untuk menjadwalkan ujian. Pertama, metode pewarnaan graf akan dipakai untuk menghasilkan solusi awal. Selanjutnya, solusi awal ini diperbaiki menggunakan Algoritma *Great Deluge Hyper-Heuristic*. Berikut adalah tahapan penjadwalan ujian.

4.3.1. Pembentukan Solusi Awal dengan Algoritma *Graph Colouring*

Algoritma *Graph Colouring* untuk memecahkan masalah penjadwalan ujian memiliki langkah awal yaitu merepresentasikan masalah dalam bentuk graf. Dalam penelitian ini, banyaknya simpul graf direpresentasikan oleh banyaknya mata kuliah yang diujiankan. Pada data York Mills Collegiate Institute terdapat 181 mata kuliah, sehingga apabila masalah ujian direpresentasikan dalam graf, maka akan ada 181 simpul graf. Jika ada dua mata kuliah yang saling bentrok maka akan dihubungkan oleh sebuah sisi. Banyak sisi dari suatu mata kuliah dalam penelitian adalah banyaknya mata kuliah yang bentrok dengan mata kuliah tersebut. Gambar 4.5 merupakan potongan tampilan program untuk list mata kuliah yang saling bentrok. Dalam gambar terdapat dua buah array dimana array pertama merupakan barisan mata kuliah yang bentrok dengan mata kuliah 1, dan array kedua merupakan barisan mata kuliah yang bentrok dengan mata kuliah 2.

```
[0, 3, 7, 9, 10, 11, 12, 13, 14, 15, 17, 19, 26, 31, 32, 33, 35, 36, 38, 50, 53, 56, 61, 62, 63, 68, 77, 81, 82, 83, 85,
 91, 94, 103, 104, 111, 112, 113, 114, 116, 117, 118, 121, 125, 127, 133, 134, 135, 136, 137, 138, 142, 143, 147, 154, 1
55, 161, 167, 169, 174, 175, 177, 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, 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, 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, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[1, 7, 10, 11, 13, 15, 17, 19, 20, 21, 32, 33, 35, 36, 38, 42, 49, 61, 62, 63, 66, 81, 82, 83, 86, 91, 94, 95, 100, 101,
 103, 104, 106, 108, 113, 114, 117, 118, 121, 122, 125, 126, 128, 133, 134, 138, 142, 143, 148, 154, 155, 160, 161, 162,
 168, 169, 170, 171, 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, 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, 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, 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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Gambar 4.5 Tampilan Program Data Konflik Mata Kuliah

Setiap mata kuliah akan diwarnai atau akan diberi *timeslot* yang berbeda dengan mata kuliah yang bentrok dengannya. Dalam proses pewarnaan simpul, penulis memakai metode *Largest Enrollment*, yaitu menjadwalkan terlebih dahulu mata kuliah yang diujiankan yang diambil oleh mahasiswa terbanyak. Karena dalam penelitian ini mata kuliah yang memiliki mahasiswa terbanyak adalah mata kuliah 40, maka pemberian warna atau *timeslot* dimulai dari mata kuliah 40, dilanjutkan oleh mata kuliah dengan mahasiswa terbanyak selanjutnya. Gambar 4.6 merupakan potongan tampilan program untuk proses pemberian warna (*timeslot*) tiap mata kuliah. Pada baris 1, terdapat tulisan "Timeslot[39] =1" yang memiliki arti bahwa mata kuliah 39+1 atau 40 dimasukkan ke dalam *timeslot* 1.

```
Timeslot[ 39 ] = 1
Timeslot[ 101 ] = 2
Timeslot[ 42 ] = 1
Timeslot[ 117 ] = 2
```

Gambar 4.6 Tampilan Program Pemberian *Timeslot* Mata Kuliah

Jika setiap mata kuliah sudah memiliki *timeslot* (terjadwalkan), maka dapat dihitung nilai *proximity*nya. Hasil penjadwalan serta nilai *proximity* dari metode *Graph Colouring* oleh program dapat dilihat pada Gambar 4.7.

```
Solusi Awal :
[18, 9, 7, 16, 20, 6, 11, 19, 11, 12, 10, 10, 14, 14, 17, 4, 8, 2, 10, 21, 13, 7, 5, 12, 11, 5, 20, 1, 1, 1, 1, 1, 3, 1,
 3, 1, 3, 18, 15, 1, 1, 18, 1, 23, 19, 2, 23, 14, 9, 2, 5, 6, 9, 15, 5, 17, 6, 13, 8, 13, 7, 5, 17, 7, 16, 17, 22, 5, 16
, 8, 8, 8, 18, 8, 10, 8, 21, 5, 5, 5, 5, 11, 11, 11, 8, 12, 18, 7, 16, 17, 15, 22, 4, 14, 19, 6, 3, 12, 13, 3, 6, 2, 4,
14, 17, 17, 7, 3, 7, 3, 3, 7, 17, 4, 4, 4, 22, 2, 13, 9, 9, 12, 10, 4, 21, 8, 5, 9, 19, 4, 4, 11, 4, 5, 5, 16, 9, 16, 7,
 16, 7, 18, 8, 8, 15, 11, 5, 5, 20, 5, 9, 2, 2, 2, 6, 6, 8, 14, 12, 10, 18, 15, 18, 15, 17, 12, 11, 16, 12, 20, 18, 23,
6, 9, 13, 15, 11, 11, 20, 21, 22]
Nilai Proximity Solusi Awal : 176837
```

Gambar 4.7 Solusi Awal Penjadwalan YOR83

Penjadwalan solusi awal yang dilakukan program dapat diuraikan perhari sebagai berikut.

Tabel 4.3 Hasil Penjadwalan Solusi Awal

Hari	*Timeslot*	Mata Kuliah
1	1	28, 29, 30, 31, 32, 34, 36, 40, 41, 43, 7, 9, 25, 82, 83, 84, 132, 146, 167
	2	18, 46, 50, 102, 118, 152, 153, 154
	3	33, 35, 37, 97, 100, 108, 110, 111
2	4	16, 93, 103, 114, 115, 116, 124, 130, 131, 133
	5	23, 26, 51, 55, 62, 68, 78, 79, 80, 81, 127, 134, 135, 147, 148, 150
	6	6, 52, 57, 96, 101, 155, 156, 173
3	7	3, 22, 61, 64, 88, 107, 109, 112, 139, 141
	8	17, 59, 70, 71, 72, 74, 76, 85, 126, 143, 144, 157
	9	2, 49, 53, 120, 121, 128, 137, 151, 174
4	10	11, 12, 19, 75, 123, 160
	11	7, 9, 25, 82, 83, 84, 132, 146, 167, 177, 178
	12	10, 24, 86, 98, 122, 159, 166, 169
5	13	21, 58, 60, 99, 119, 175
	14	13, 14, 48, 94, 104, 158
	15	39, 54, 91, 145, 162, 164, 176
6	16	4, 65, 69, 89, 136, 138, 140, 168
	17	15, 56, 63, 66, 90, 105, 106, 113, 165
	18	1, 38, 42, 73, 87, 142, 161, 163, 171
7	19	8, 45, 95, 129
	20	5, 27, 149, 170, 179
	21	20, 77, 125, 180
8	22	67, 92, 117, 181
	23	44, 47, 172

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

Solusi awal yang telah dibuat menggunakan metode *Graph Colouring* kemudian akan diperbaiki oleh algoritma *Great Deluge Hyper-Heuristic*. Berikut adalah parameter yang digunakan untuk inisialisi awal algoritma *Great Deluge Hyper-Heuristic*:

$fs = 176837$

$D = 181$ (diambil nilai *proximity* yang mendekati 0)

$I = 1000$

$B = fs = 176837$

$\Delta B = \frac{fs-D}{I} = \frac{176837-181}{1000} = 176.656$

Solusi yang telah dihasilkan algoritma *Graph Colouring* akan diubah untuk menemukan jadwal yang lebih baik dengan menggunakan dua pendekatan *low level heuristic*. Pada setiap iterasi, akan ada pemilihan *low level heuristic* yang darinya menghasilkan kandidat jadwal baru. Kemudian nilai *proximity* 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*.

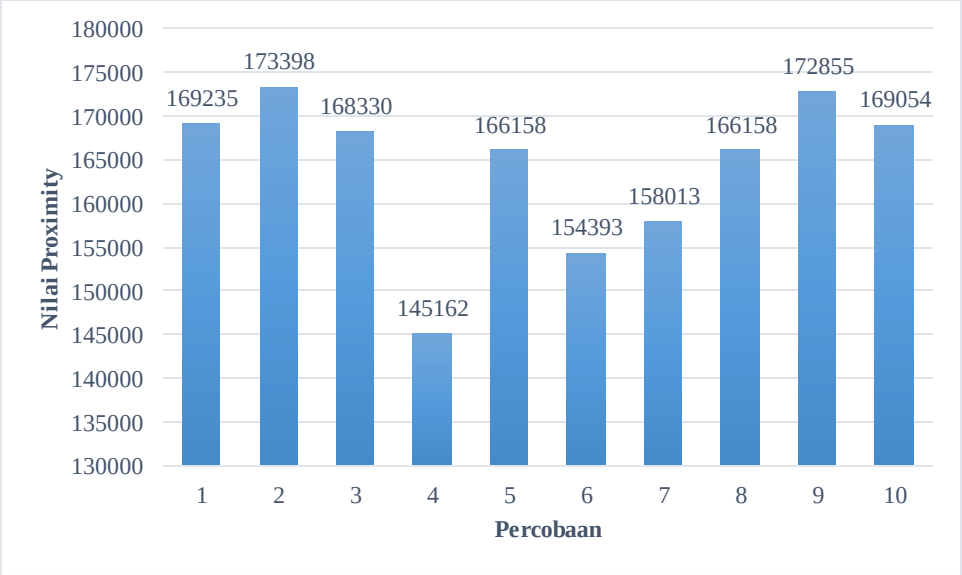
4.4. Analisis Parameter

Algoritma *Great Deluge Hyper-Heuristic* digunakan untuk melakukan perbaikan pada jadwal yang telah dibuat oleh algoritma *Graph Colouring*. Sebelum mengimplementasikan Algoritma *Great Deluge Hyper-Heuristic* pada masalah penjadwalan ujian perlu dilakukan analisis parameter algoritma *Great Deluge Hyper-Heuristic*. Analisis parameter dilakukan supaya mendapatkan pengaturan nilai parameter yang tepat agar algoritma dapat berjalan secara efisien dan dapat menghasilkan solusi yang optimal. Dalam penelitian ini, terdapat parameter yang digunakan yaitu banyak iterasi (I). Banyak iterasi pada algoritma *Great Deluge Hyper-Heuristic* tersebut dianalisis pengaruhnya terhadap nilai *proximity*. Uji coba dilakukan sebanyak sepuluh kali dikarenakan algoritma *Great Deluge* merupakan algoritma yang bersifat stokastik sehingga tidak dapat menghasilkan solusi yang pasti.

Dalam penelitian ini, penulis melakukan uji coba dengan mengubah nilai banyak iterasi menjadi 200, 400, 600, 800, dan 1000.

1. Iterasi 200

Pada uji coba ini nilai parameter I diubah menjadi 200 yang memiliki arti bahwa algoritma melakukan iterasi sebanyak 200 kali.

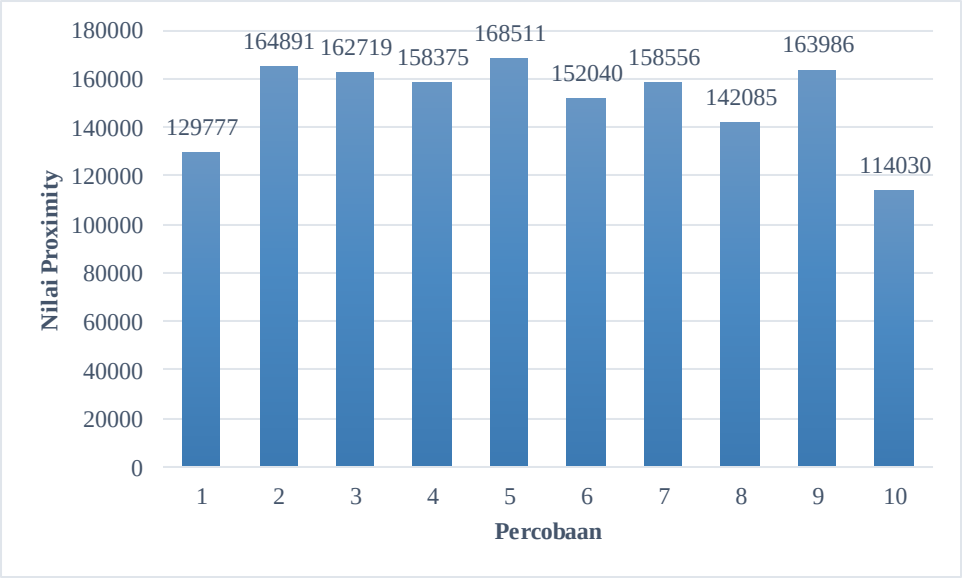


Gambar 4.8 Grafik Nilai *Proximity* Iterasi 200

Dari Gambar 4.8, didapatkan nilai *proximity* sebesar 145162 sebagai nilai *proximity* terbaik dan 164275.6 sebagai rata-rata nilai *proximity* uji coba iterasi 200 ini.

2. Iterasi 400

   Pada uji coba ini algoritma melakukan iterasi sebanyak 400 kali. Nilai *proximity* dari hasil uji coba iterasi 400 dapat dilihat pada Gambar 4.9.

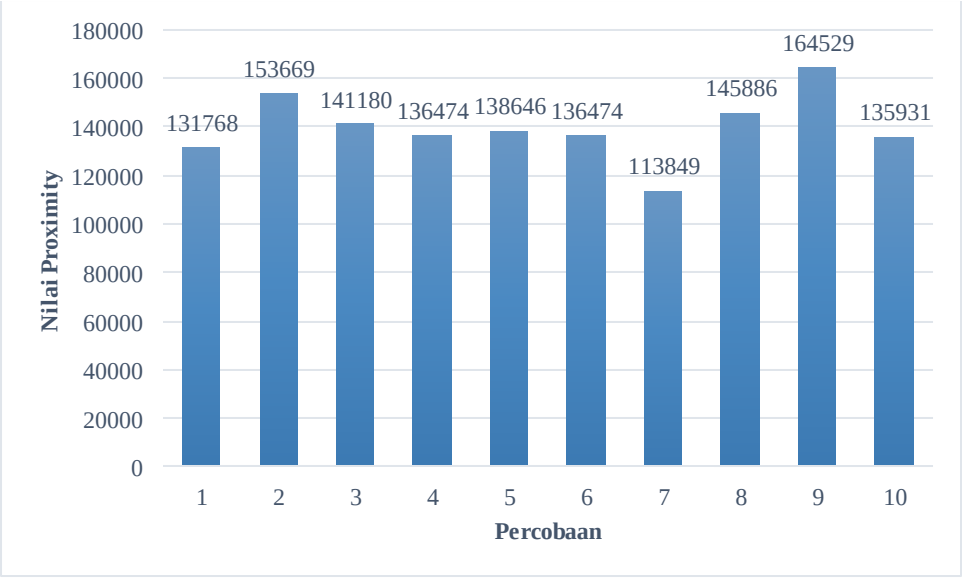


Gambar 4.9 Grafik Nilai *Proximity* Iterasi 400

Dari sepuluh kali percobaan, didapatkan nilai *proximity* terbaik yaitu 114030 pada percobaan kesepuluh dan rata-rata nilai *proximity* uji coba iterasi 400 ini adalah 151497. Rata-rata nilai *proximity* yang didapat dari percobaan ini lebih baik dibandingkan dengan rata-rata percobaan iterasi 200 sebelumnya. Sehingga iterasi 400 lebih baik dibandingkan iterasi 200.

3. Iterasi 600

Pada uji coba ini nilai parameter I diubah menjadi 600 untuk melihat pengaruhnya terhadap nilai *proximity*.

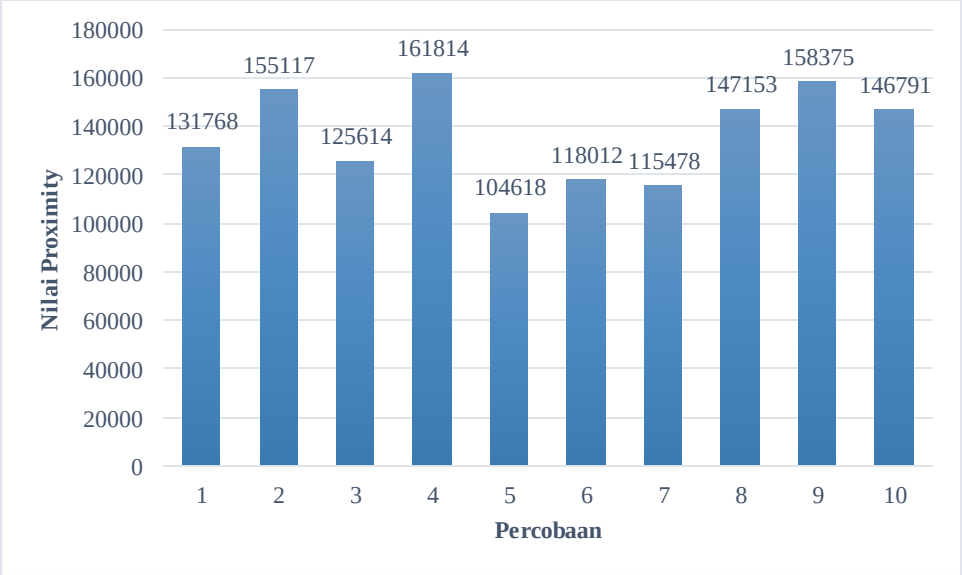


Gambar 4.10 Grafik Nilai *Proximity* Iterasi 600

Dari Gambar 4.10, didapatkan nilai *proximity* sebesar 113849 sebagai nilai *proximity* terbaik dan rata-rata nilai *proximity* uji coba iterasi 600 ini adalah 139840.6. Karena rata-rata nilai *proximity* pada percobaan ini lebih baik dibandingkan rata-rata nilai *proximity* pada percobaan iterasi 400, maka iterasi 600 lebih baik dibandingkan iterasi 400.

4. Iterasi 800

Pada uji coba ini nilai parameter I diubah menjadi 800 untuk melihat pengaruhnya terhadap nilai *proximity*. Hasil percobaan dapat dilihat pada gambar berikut.

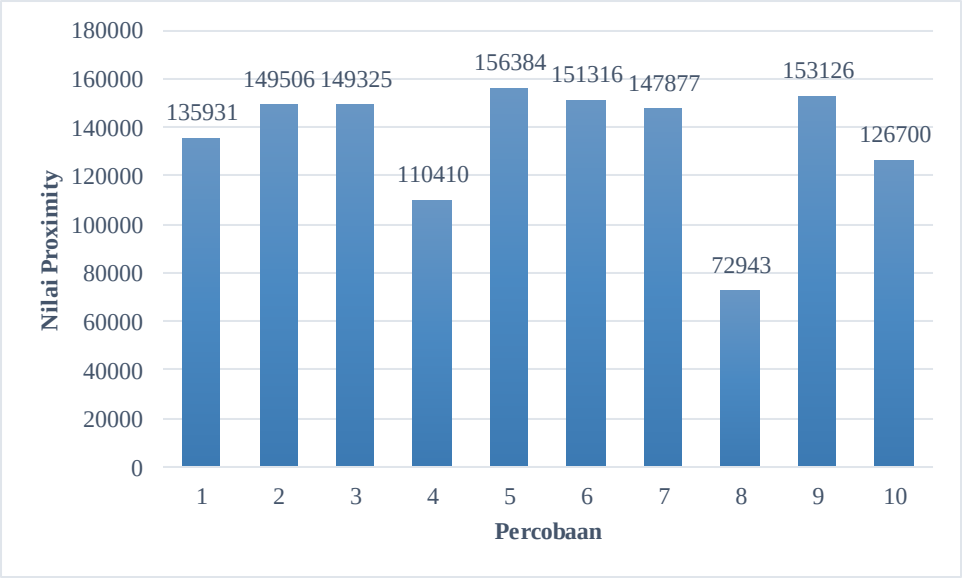


Gambar 4.11 Grafik Nilai *Proximity* Iterasi 800

Dari hasil sepuluh kali percobaan, didapatkan nilai *proximity* sebesar 104618 sebagai nilai *proximity* terbaik dan 136474 sebagai rata-rata nilai *proximity* uji coba iterasi 800 ini. Rata-rata pada percobaan ini lebih rendah dibandingkan percobaan sebelumnya, hal tersebut menunjukkan bahwa iterasi 800 lebih baik dibandingkan iterasi 600.

5. Iterasi 1000

Pada uji coba ini nilai parameter I diubah menjadi 1000.



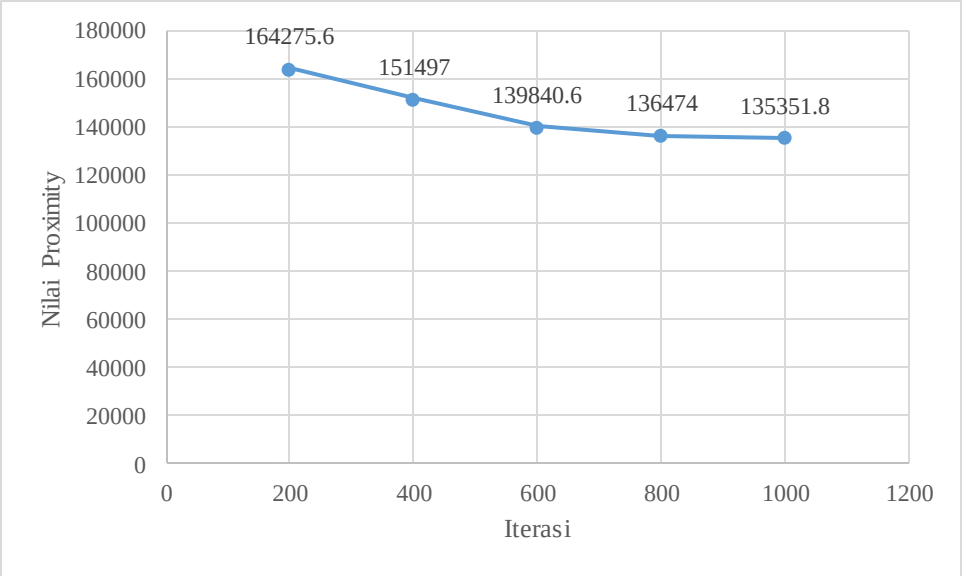
Gambar 4.12 Grafik Nilai *Proximity* Iterasi 1000

Dari Gambar 4.12, didapatkan nilai *proximity* sebesar 72943 sebagai nilai *proximity* terbaik dan 135351.8 sebagai rata-rata nilai *proximity* uji coba iterasi 1000 ini. Karena rata-rata nilai *proximity* pada percobaan ini lebih baik dibandingkan rata-rata nilai *proximity* pada percobaan iterasi 800, maka iterasi 1000 lebih baik dibandingkan iterasi 800.

Selengkapnya, perbandingan antara nilai *proximity* tiap iterasi diuraikan pada Tabel 4.4

Tabel 4.4 Perbandingan Hasil Uji Coba Iterasi

	Iterasi				
Percobaan	200	400	600	800	1000
1	169235	129777	131768	131768	135931
2	173398	164891	153669	155117	149506
3	168330	162719	141180	125614	149325
4	145162	158375	136474	161814	110410
5	166158	168511	138646	104618	156384
6	154393	152040	136474	118012	151316
7	158013	158556	113849	115478	147877
8	166158	142085	145886	147153	72943
9	172855	163986	164529	158375	153126
10	169054	114030	135931	146791	126700
Max	173398	168511	164529	161814	156384
Min	145162	114030	113849	104618	72943
Rata-Rata	164275.6	151497	139840.6	136474	135351.8



Gambar 4.13 Grafik Perbandingan Nilai *Proximity* Setiap Iterasi

Dari hasil percobaan, didapatkan bahwa iterasi dengan jumlah 1000 menghasilkan nilai rata-rata yang paling baik. Nilai *proximity* dengan nilai 72943 merupakan nilai *proximity* terbaik dari 40 kali percobaan. Kemudian dapat dilihat pada Gambar 4.13, semakin besar banyak iterasi, maka semakin kecil rata-rata nilai *proximity*nya. Jadi dapat disimpulkan bahwa banyak iterasi di dalam algoritma *Great Deluge* mempengaruhi nilai *proximity*. Hasil jadwal dengan fungsi tujuan terbaik dari setiap iterasi dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Penjadwalan Nilai *Proximity* Terbaik Tiap Iterasi

Banyak Iterasi	Nilai *Proximity*	Usulan Jadwal

200	145162	18, 9, 7, 16, 20, 6, 11, 19, 11, 12, 10, 10, 14, 14, 17, 4, 8, 2, 20, 21, 13, 7, 5, 12, 11, 5, 20, 1, 1, 1, 1, 1, 3, 1, 3, 1, 3, 18, 15, 1, 1, 18, 1, 23, 19, 2, 23, 14, 9, 2, 5, 6, 9, 15, 5, 21, 6, 13, 8, 13, 7, 5, 17, 7, 16, 17, 22, 5, 16, 8, 8, 8, 18, 8, 10, 8, 21, 5, 5, 5, 5, 11, 11, 11, 8, 12, 18, 7, 16, 17, 15, 22, 4, 14, 19, 6, 3, 12, 13, 3, 6, 2, 4, 14, 17, 17, 7, 21, 7, 3, 3, 7, 17, 4, 4, 4, 22, 22, 13, 9, 9, 12, 23, 4, 21, 8, 5, 9, 19, 4, 4, 11, 4, 5, 21, 16, 9, 16, 7, 16, 7, 18, 8, 8, 15, 11, 5, 5, 20, 5, 9, 2, 2, 2, 8, 6, 8, 14, 12, 10, 18, 15, 18, 15, 17, 12, 11, 16, 18, 20, 23, 22, 6, 9, 13, 15, 11, 21, 20, 21, 2
400	114030	18, 9, 7, 9, 20, 16, 20, 19, 11, 12, 10, 10, 14, 14, 17, 4, 8, 2, 10, 21, 20, 7, 5, 12, 11, 7, 20, 1, 1, 1, 1, 1, 3, 1, 3, 1, 23, 18, 15, 2, 17, 18, 1, 23, 19, 3, 23, 14, 9, 2, 5, 6, 22, 15, 5, 17, 6, 13, 8, 13, 7, 5, 17, 10, 16, 17, 22, 5, 13, 8, 8, 8, 18, 8, 10, 8, 21, 5, 5, 21, 19, 11, 11, 11, 8, 12, 18, 7, 16, 17, 15, 22, 5, 14, 19, 19, 3, 12, 13, 3, 6, 2, 4, 14, 17, 17, 7, 3, 7, 3, 22, 7, 17, 4, 4, 4, 22, 2, 13, 14, 9, 12, 10, 4, 21, 22, 5, 9, 19, 4, 4, 11, 4, 5, 5, 16, 9, 16, 7, 16, 7, 18, 8, 8, 15, 11, 5, 5, 20, 5, 8, 2, 23, 2, 6, 6, 8, 14, 15, 14, 18, 15, 18, 15, 17, 12, 11, 16, 12, 23, 18, 6, 4, 23, 13, 15, 11, 11, 20, 16, 18

600	113849	18, 9, 7, 16, 20, 6, 11, 19, 11, 12, 10, 10, 14, 14, 17, 4, 8, 2, 10, 21, 13, 7, 10, 12, 11, 17, 20, 1, 1, 1, 1, 1, 3, 1, 3, 23, 3, 18, 15, 22, 1, 18, 1, 23, 19, 14, 23, 14, 9, 2, 21, 6, 9, 15, 5, 17, 6, 13, 8, 13, 7, 5, 17, 7, 16, 17, 22, 5, 6, 8, 8, 8, 18, 8, 10, 8, 21, 9, 5, 5, 5, 11, 11, 11, 8, 12, 18, 7, 16, 17, 15, 22, 4, 14, 19, 23, 6, 12, 13, 3, 6, 2, 4, 14, 17, 17, 7, 3, 7, 22, 3, 7, 17, 4, 4, 4, 22, 2, 13, 14, 16, 23, 10, 4, 21, 8, 5, 9, 19, 4, 4, 11, 19, 5, 5, 16, 15, 16, 11, 16, 7, 18, 8, 8, 15, 21, 5, 5, 20, 5, 9, 15, 23, 2, 6, 13, 8, 14, 21, 10, 18, 15, 18, 11, 17, 12, 11, 16, 12, 23, 18, 22, 6, 9, 13, 15, 11, 11, 20, 17, 23
800	104618	18, 9, 7, 16, 20, 6, 11, 19, 11, 12, 10, 10, 14, 14, 17, 23, 22, 2, 10, 21, 13, 7, 12, 12, 11, 17, 20, 16, 1, 1, 10, 1, 1, 3, 3, 3, 3, 18, 15, 1, 17, 18, 6, 23, 19, 19, 23, 14, 9, 2, 21, 13, 9, 15, 5, 17, 9, 13, 8, 13, 7, 5, 17, 7, 19, 17, 22, 5, 13, 6, 8, 8, 18, 8, 10, 8, 21, 5, 22, 22, 22, 11, 6, 11, 8, 22, 18, 7, 16, 17, 15, 22, 4, 14, 19, 11, 22, 12, 13, 3, 6, 2, 4, 14, 17, 17, 7, 14, 7, 3, 3, 7, 17, 4, 4, 4, 22, 2, 13, 9, 9, 12, 16, 4, 21, 8, 5, 9, 19, 19, 4, 11, 19, 5, 5, 16, 9, 16, 7, 16, 7, 18, 8, 8, 15, 11, 15, 23, 20, 5, 8, 2, 2, 2, 8, 4, 8, 14, 12, 20, 18, 15, 18, 15, 17, 23, 11, 16, 16, 20, 23, 4, 4, 9, 13, 15, 11, 21, 4, 16, 6

1000	72943	18, 9, 7, 16, 20, 21, 11, 19, 11, 12, 10, 10, 14, 14, 17, 4, 8, 2, 20, 21, 23, 7, 5, 12, 11, 5, 20, 1, 16, 1, 22, 1, 3, 1, 3, 4, 3, 18, 13, 1, 1, 18, 1, 23, 19, 2, 4, 14, 9, 2, 9, 6, 9, 23, 5, 21, 13, 6, 8, 13, 7, 13, 8, 10, 16, 17, 22, 5, 6, 21, 17, 8, 18, 8, 10, 22, 21, 5, 5, 16, 5, 11, 11, 11, 12, 12, 18, 4, 16, 17, 15, 22, 14, 14, 19, 6, 22, 23, 13, 3, 6, 15, 2, 14, 20, 17, 7, 3, 7, 3, 19, 7, 17, 15, 4, 4, 22, 17, 13, 19, 9, 12, 10, 20, 15, 20, 5, 9, 19, 4, 4, 11, 4, 5, 5, 16, 23, 8, 7, 16, 7, 22, 8, 8, 15, 11, 6, 5, 20, 5, 6, 2, 18, 20, 8, 12, 8, 14, 12, 23, 18, 8, 7, 15, 17, 2, 11, 16, 12, 21, 6, 11, 23, 10, 13, 20, 9, 12, 20, 16, 13

4.5. Hasil Implementasi

Solusi awal untuk penjadwalan ujian di York Mills Collegiate Institute menggunakan Graph Colouring dapat dilihat pada Gambar 4.7. Dari hasil penjadwalan tersebut diperoleh nilai *proximity* solusi awal yaitu 176837. Solusi awal yang telah dibuat menggunakan metode *Graph Colouring* merupakan solusi yang tidak melanggar *hard constraints*, akan tetapi masih banyak melanggar *soft constraints*. Oleh karena itu, solusi awal kemudian akan diperbaiki oleh algoritma *Great Deluge Hyper-Heuristic* agar mengurangi pelanggaran terhadap *soft constraints*. Berikut adalah hasil penjadwalan ujian di York Mills Collegiate Institute menggunakan algoritma *Great Deluge Hyper-Heuristic*. Hasil penjadwalan diperoleh dengan mengambil iterasi terbaik dari analisis parameter yaitu 1000. Peneliti mengambil solusi yang menghasilkan nilai *proximity* terkecil yang dilakukan saat uji coba iterasi 1000 yaitu jadwal yang memiliki nilai *proximity* 72943. Hasil penjadwalan program dapat dilihat pada gambar berikut.

```
Jadwal terbaik setelah 1000 iterasi:
[18, 9, 7, 16, 20, 21, 11, 19, 11, 12, 10, 10, 14, 14, 17, 4, 8, 2, 20, 21, 23, 7, 5, 12, 11, 5, 20, 1, 16, 1, 22, 1, 3,
 1, 3, 4, 3, 18, 13, 1, 1, 18, 1, 23, 19, 2, 4, 14, 9, 2, 9, 6, 9, 23, 5, 21, 13, 6, 8, 13, 7, 13, 8, 10, 16, 17, 22, 5,
 6, 21, 17, 8, 18, 8, 10, 22, 21, 5, 5, 16, 5, 11, 11, 11, 12, 12, 18, 4, 16, 17, 15, 22, 14, 14, 19, 6, 22, 23, 13, 3,
6, 15, 2, 14, 20, 17, 7, 3, 7, 3, 19, 7, 17, 15, 4, 4, 22, 17, 13, 19, 9, 12, 10, 20, 15, 20, 5, 9, 19, 4, 4, 11, 4, 5,
5, 16, 23, 8, 7, 16, 7, 22, 8, 8, 15, 11, 6, 5, 20, 5, 6, 2, 18, 20, 8, 12, 8, 14, 12, 23, 18, 8, 7, 15, 17, 2, 11, 16,
12, 21, 6, 11, 23, 10, 13, 20, 9, 12, 20, 16, 13]
Nilai proximity jadwal terbaik: 72943
```

Gambar 4.14 Tampilan *Software* Hasil Penjadwalan Algoritma *Great Deluge Hyper-Heuristic*

Dari hasil penjadwalan, karena setiap mata kuliah sudah mendapatkan *timeslot*nya masing-masing, maka dapat dibuat list untuk *timeslot* apa saja yang diambil oleh setiap mahasiswa beserta hari apa saja mahasiswa melakukan ujian. Pada Gambar 4.15, setiap baris menerangkan barisan *timeslot* ujian setiap mahasiswa sedangkan angka dalam barisan menerangkan hari ujian. Misal, baris 1 menjelaskan bahwa mahasiswa 1 dijadwalkan ujian pada *timeslot* 1 dan *timeslot* 15. Sedangkan pada Gambar 4.16, setiap baris menerangkan barisan hari ujian setiap mahasiswa sedangkan angka dalam barisan menerangkan hari ujian. Apabila dalam suatu baris terdapat lebih dari satu angka sama maka memiliki arti bahwa terdapat lebih dari satu ujian yang dijadwalkan pada angka/hari tersebut. Misal, baris 1 menjelaskan bahwa mahasiswa 1 dijadwalkan ujian pada hari ke-1 dan hari ke-5.

```
[1, 15]
[11, 6, 15, 20]
[3, 6, 11, 20, 15, 12, 5, 12]
[10, 4, 2, 1, 12, 20, 5, 18]
[3, 2, 17, 17, 10, 5, 14, 6]
[1, 6, 4]
[16, 19, 14, 8, 3, 7]
[4, 6, 15, 15, 6]
[3, 13, 7, 5, 6, 11, 9]
[1, 10, 11, 19, 15, 12, 16, 8]
```

Gambar 4.15 Potongan Tampilan *Software Timeslot* Setiap Mahasiswa

```
[1, 5]
[2, 4, 5, 7]
[1, 2, 2, 4, 4, 4, 5, 7]
[1, 1, 2, 2, 4, 4, 6, 7]
[1, 1, 2, 2, 4, 5, 6, 6]
[1, 2, 2]
[1, 3, 3, 5, 6, 7]
[2, 2, 2, 5, 5]
[1, 2, 2, 3, 3, 4, 5]
[1, 3, 4, 4, 4, 5, 6, 7]
```

Gambar 4.16 Potongan Tampilan *Software* Hari Ujian Setiap Mahasiswa

Selengkapnya, jadwal yang dihasilkan algoritma *Great Deluge Hyper-Heuristic* dapat diuraikan pada Tabel 4.6.

Tabel 4.6 Hasil Penjadwalan Algoritma *Great Deluge Hyper-Heuristic*

Hari	*Timeslot*	Mata Kuliah
1	1	28, 30, 32, 34, 40, 41, 43
	2	18, 46, 50, 103, 152, 166
	3	33, 35, 37, 100, 108, 110
2	1	16, 36, 47, 88, 115, 116, 130, 131, 133
	2	23, 26, 55, 68, 78, 79, 81, 127, 134, 135, 148, 150
	3	52, 58, 69, 96, 101, 147, 151, 171
3	1	3, 22, 61, 107, 109, 112, 139, 141, 163
	2	17, 59, 63, 72, 74, 138, 143, 144, 155, 157, 162
	3	2, 49, 51, 53, 121, 128, 177
4	1	11, 12, 64, 75, 123, 174
	2	7, 9, 25, 82, 83, 84, 132, 146, 167, 172
	3	10, 24, 85, 86, 122, 156, 159, 169, 178
5	1	39, 57, 60, 62, 99, 119, 175, 181
	2	13, 14, 48, 93, 94, 104, 158
	3	91, 102, 114, 125, 145, 164
6	1	4, 29, 65, 80, 89, 136, 140, 168, 180
	2	15, 66, 71, 90, 106, 113, 118, 165
	3	1, 38, 42, 73, 87, 153, 161
7	1	8, 45, 95, 111, 120, 129
	2	5, 19, 27, 105, 124, 126, 149, 154, 176, 179
	3	6, 20, 56, 70, 77, 170
8	1	31, 67, 76, 92, 97, 117, 142
	2	21, 44, 54, 98, 137, 160, 173

Berdasarkan hasil penjadwalan yang dilakukan oleh algoritma *Great Deluge*, terdapat penurunan nilai *proximity* dari solusi awal. Jadwal solusi awal memiliki nilai *proximity* sebesar 176837 dan jadwal setelah diperbaiki oleh algoritma *Great Deluge Hyper-Heuristic* adalah 72943. Hal ini menunjukkan bahwa algoritma

Great Deluge Hyper-Heuristic dapat menghasilkan jadwal ujian lebih baik karena mampu mengurangi pelanggaran terhadap *soft constraints* yang ada. Dengan demikian dapat disimpulkan bahwa metode *Great Deluge Hyper-Heuristic* dapat diimplementasikan untuk penyelesaian penjadwalan ujian.