p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019

PENDEKATAN HYPER HEURISTIC DENGAN KOMBINASI ALGORITMA PADA EXAMINATION TIMETABLING PROBLEM

Vicha Azthanty Supoyo[1], Ahmad Muklason[2]
[1]vichasupoyo@gmail.com, [2]ahmad.muklason@gmail.com
[1,2]Departemen Sistem Informasi, Fakultas Teknologi Informasi dan Komunikasi,
Institut Teknologi Sepuluh Nopember, Surabaya

Abstrak

Umumnya, penjadwalan ujian masih dilakukan secara manual dan tentu saja membutuhkan waktu yang cukup lama. Banyak penelitian yang telah mengembangkan berbagai pendekatan untuk menemukan strategi yang lebih tepat untuk masalah penyusunan jadwal. *Hyperheuristic* adalah pendekatan yang diusulkan pada penelitian ini. Pada *Hyperheuristic, Simple Random* digunakan sebagai strategi untuk memilih *low-level-heuristic* sementara *Hill Climbing* dan *Simulated Annealing* sebagai strategi *move acceptance.* Data set Carter digunakan sebagai bahan pengujian. Dalam penelitian ini, diusulkan pengujian terhadap data set dengan batas waktu 15 menit hingga 1 jam dan hasilnya dibandingkan dengan penelitian yang dilakukan oleh Carter sebagai penelitian awal yang menggunakan data set tersebut. Selain itu, juga dilakukan pengaturan data set, jumlah iterasi, dan batas waktu sama dengan salah satu literatur yang selanjutnya akan dijadikan pembanding. Hasil yang diperoleh menunjukkan satu pasang algoritma yang diusulkan pada penelitian ini lebih baik jika dibandingkan dengan literatur sedangkan algoritma lainnya juga memberikan hasil yang signifikan.
Kata kunci: *Hyperheuristic,* Data set *Carter* (Toronto), Algoritma *Simple Random*, Algoritma *Hill Climbing,* Algoritma *Simulated Annealing*

Abstract

Generally, exam scheduling is still done manually and definitely will take a long time. Many researches have developed various studies to find a more appropriate strategy. Hyperheuristic was proposed in this study. In Hyperheuristic, Simple Random is used as a strategy for selecting low-level-heuristic while Hill Climbing and Simulation Annealing as move acceptance strategy. The Carter dataset is used as a test for the algorithms. We proposed testing datasets with a time limit of 15 minutes up to 1 hour and the results were compared with the research conducted by Carter as an initial study using that dataset. In addition, dataset, the number of iterations, and the time limit are as same as one of the literatures which will then be compared. The results obtained show that one pair of algorithms proposed in this study is better than the literature while other algorithms also provide significant results.
***Keywords**:* Hyper-heuristic, Carter (Toronto) Datasets, Simple Random Algorithm, Hill Climbing Algorithm*,* Simulated Annealing Algorithm

1. Pendahuluan

Penjadwalan ujian merupakan salah satu proses dalam ruang lingkup institusi pendidikan yang termasuk dalam kategori masalah optimasi. Optimasi merupakan proses pencarian solusi terbaik, menemukan nilai fungsi objektif (minimum atau maksimum), dengan mempertimbangkan batasan atau syarat yang ditetapkan oleh masing-masing institusi. Penelitian ini mengusulkan konsep *single objective optimization* dengan tujuan untuk menginvestigasi apakah algoritma yang diusulkan mampu menemukan nilai minimum dari objective function jika dibandingkan dengan hasil yang dilaporkan pada literatur. Nilai minimum tersebut diperoleh dari hasil pengukuran kualitas setiap solusi. Pada umumnya, proses penjadwalan ujian masih dilakukan secara manual. Selain dituntut membentuk jadwal ujian dengan cepat, institusi pendidikan juga harus mempertimbangkan ketersediaan ruang dan periode pelaksanaan yang terbatas dalam kurun waktu lebih singkat jika dibandingkan dengan pelaksanaan perkuliahan. Hal yang tidak kalah penting lainnya adalah bagaimana mahasiswa atau para peserta dapat dijadwalkan mengikuti ujian tanpa adanya bentrokan antara satu dan lainnya. Dengan pertimbangan tersebut, tentunya akan dibutuhkan waktu yang cukup lama. Di lingkungan peneliti, khususnya ruang lingkup *exam timetabling*, hal-hal tersebut dikenal dengan istilah hard constraint dan soft constraint. Hard constraint adalah batasan yang harus dipenuhi sedangkan untuk batasan yang harus seminimal mungkin dilanggar disebut soft constraint. Terdapat perbedaan tentang apa saja yang dikategorikan sebagai hard constraint dan soft constraint dari setiap institusi[1].

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019

Peneliti menggunakan data set Toronto, atau biasa juga disebut dengan data set Carter pada penelitian ini. Adapun hard constraint dan soft constraint dari data set ini akan dijelaskan pada bagian selanjutnya. Data set Carter mengandung *instansi* yang sesuai dengan variabel permasalahan dari dunia nyata karena berasal dari institusi pendidikan yang berbeda dan dikategorikan sebagai data set yang stabil dan sering digunakan dalam penelitian[2]. Sejauh yang diketahui peneliti, rata-rata penelitian dengan pengujian terhadap data set tersebut akan menemukan solusi yang baik ketika dilakukan pada batas waktu 1 hingga 12 jam. Dalam penelitian ini, peneliti melakukan pengujian data set dengan batas waktu kurang dari satu jam. Selain itu, juga dilakukan pengaturan batas waktu, jumlah iterasi dan data set sama dengan salah satu literatur yang selanjutnya akan dijadikan pembanding.

Penelitian ini melakukan pengujian beberapa algoritma terhadap data set Carter (Toronto) melalui pendekatan Hyper-Heuristik. Ketika pendekatan secara tradisional fokus pada ruang solusi, pencarian dengan hyper-heuristik lebih difokuskan pada ruang heuristik [3]. Hal tersebut merujuk pada salah satu ciri khas dari hyper-heuristik yaitu pemisahan letak logika pemrosesan *problem domain* dan metodologi. Bagian metodologi terletak pada level tertinggi, artinya komponen yang terdapat pada level ini tidak terpengaruh lagi bahkan ketika *problem domain* diubah. Konsep ini bertujuan untuk membangun atau menemukan metode optimasi yang sifatnya general sehingga dapat dengan mudah diimplementasikan pada masalah optimasi apapun.

Hyper-heuristik bekerja seperti black box systems dimana data yang digunakan hanya berupa *non-problem-specific* data yang telah disediakan oleh setiap low-level heuristic (llh) dengan tujuan untuk memilih llh dan menggunakan llh tersebut kepada solusi kandidat[4]. Lebih rinci, penelitian ini akan fokus pada metodologi untuk memilih strategi heuristic dengan dua tahapan di dalamnya yaitu *heuristic selection* dan *move acceptance*[5].

Simple Random-Hill Climbing dan *Simple Random-Simulated Annealing* yang digunakan pada penelitian kami. Dimana *Simple Random* akan bertugas pada bagian *heuristic selection* sedangkan *Hill Climbing* dan *Simulated Annealing* sebagai *move acceptance.* Sedangkan daftar low-level heuristic yang digunakan akan dijelaskan lebih lanjut.

Bagian lain dari penelitian ini akan dijelaskan kedalam beberapa bagian sebagai berikut: Bagian 2 memberikan rincian Metode Penelitian, Bagian 3 akan menampilkan Hasil dan Pembahasan, dan Bagian 4 terdiri dari Kesimpulan dan Saran.

2. Metode

2.1. Deskripsi Problem Domain

Jika membahas data set dalam lingkup pendidikan, terdapat salah satu data set yang sering digunakan oleh para peneliti. Data set tersebut adalah Toronto sebagai data set yang dikenalkan oleh Carter dkk (1996). Data set Carter atau Toronto mengandung *instansi* yang sesuai dengan variabel permasalahan dunia nyata, dari institusi pendidikan yang berbeda dan dikategorikan sebagai data set yang stabil dan sering digunakan dalam penelitian [2]. Instansi pada data set Carter dapat dilihat pada Tabel 1.

Exam timetabling problem pada data set toronto dapat didefinisikan sebagai alokasi ujian ke dalam satu set slot waktu yang dapat ditempati ketika memenuhi satu set *constraint* (*hard* dan *soft*) pada saat yang bersamaan. Berikut ini *constraint* yang terdapat pada *data set* Toronto:

a) Hard Constraint: Tidak boleh ada siswa yang menempati lebih dari satu ujian pada periode waktu yang sama, atau dengan kata lain dua ujian yang memiliki peserta ujian yang sama tidak boleh dijadwalkan pada periode yang sama.
b) Soft Constraint: Mengalokasikan ujian yang memiliki konflik dalam rentang waktu berjauhan sehingga siswa dapat memiliki lebih banyak waktu untuk melakukan revisi atau perbaikan.

Ketika hard constraint dipenuhi selanjutnya digunakan fungsi objektif untuk meminimalisir total pinalti. Pinalti diperoleh tergantung dari seberapa besar pelanggaran soft constraint. Berikut ini adalah fungsi objektif untuk menghitung pinalti dapat dilihat pada persamaan (1):

$$fitness = \frac{\left(\sum_{s=0}^{4} \omega_s \times N_s\right)}{S} \qquad (1)$$

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

Tabel 1. Data Set Carter

Problem Instance	Time Slots	Exams	Students	Conflict Density
CAR91	35	682	16925	0.13
CAR92	32	543	18419	0.14
EAR83	24	190	1125	0.27
HEC92	18	81	2823	0.42
KFU93	20	461	5349	0.06
LSE91	18	381	2726	0.06
PUR93	42	2419	30029	0.03
RYE92	23	486	11483	0.07
STA83	13	139	611	0.14
TRE92	23	261	4360	0.18
UTA92	35	622	21266	0.13
UTE92	10	184	2749	0.08
YOR83	21	181	941	0.29

Huruf *s* kecil menandakan timeslot dari dua examination, yang dipilih oleh siswa, yang letaknya berjauhan satu sama lain. Jarak kedua timeslot tersebut dihitung dengan bobot $\omega_s = 2^s$. Misalnya timeslot s = 0, 1, 2, 3, 4 merupakan nilai bobot untuk exam terjadwal yang memiliki jarak, ditulis berurutan sesuai dengan bobot timeslot, yaitu 4, 3, 2, 1, 0. Sebagai contoh: seorang siswa memiliki dua ujian, a dan b, dimana keduanya terpisah pada 3 rentang timeslot. Sehingga bobot penalty timeslot tersebut adalah $\omega_s = 2^1$, maka $\omega_s = 2$. Sedangkan N_s representasi dari jumlah siswa yang melanggar soft constraint. Dan *S* adalah total siswa di dalam penjadwalan [6].

Pada komunitas peneliti sendiri, khususnya di dalam lingkup Exam Timetabling Problem, sekumpulan data yang ditawarkan oleh Michael Carter merupakan salah satu contoh koleksi data yang dibuka untuk umum. Data set tersebut dapat didownload pada link berikut ini http://www.cs.nott.ac.uk/~pszrq/data.htm.

2.2. Algoritma dengan Pendekatan Hyper Heuristic

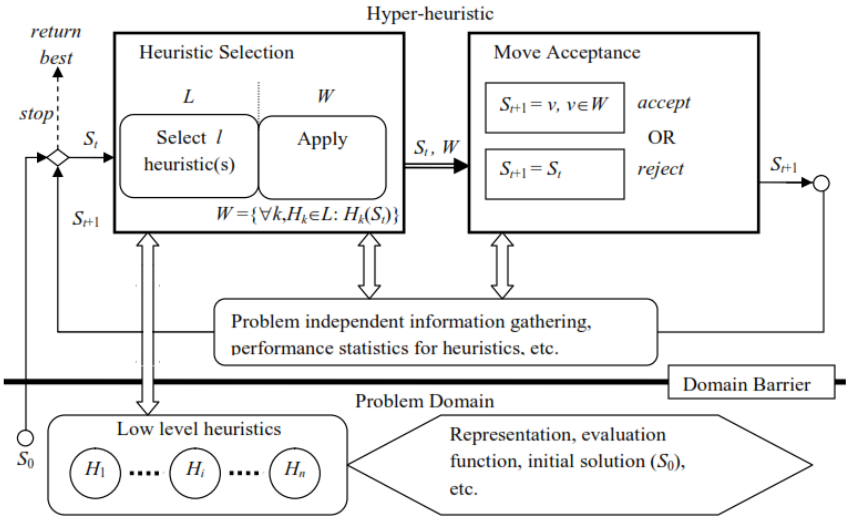
Beberapa survei *state-of-the-art* dari formulasi exam timetabling problem, disingkat sebagai ETP, dimulai dari algoritma tradisional *graph colouring*, hingga *meta-heuristic* dan *hyper-heuristic* telah dilakukan pada beberapa penelitian[7]. Pendekatan meta-heuristic, disingkat sebagai MH, telah sukses diaplikasikan untuk memecahkan berbagai masalah optimasi dengan kombinasi yang kompleks termasuk dalam pengembangan *state-of-the-art* pada area penjadwalan. Meskipun demikian, pendekatan tersebut tidak bisa dengan mudah diaplikasikan pada masalah optimasi lainnya. Hal tersebut dikarenakan, sifat MH pada umumnya mengatur parameter dan proses di dalamnya sesuai dan khusus untuk sebuah masalah saja[8]. Sehingga akan terjadi perubahan besar-besaran ketika problem domain diubah.

Sebagaimana dapat dilihat pada Gambar 1, Hyper-heuristic (HH) diperkenalkan sebagai metode optimisasi secara umum, mampu menjelajahi heuristic space daripada langsung ke solution space[3]. HH dijalankan untuk mendukung algoritma tradisional dari heuristic.

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019



Gambar 1. Demonstrasi Framework Hyper Heuristic Secara Umum[9]

Idenya dapat dilakukan dengan mengembangkan kelebihan dari setiap algoritma atau mengkombinasikan dua atau lebih algoritma, dikenal dengan sebutan hybrid metaheuristic. Pada metaheuristic standar and hybrid metaheuristic, proses berfokus pada ruang pencarian solusi untuk sebuah masalah. Perbedaannya terletak pada jumlah strategi heuristik yang yang digunakan. Pada metaheuristik standar, strategi yang digunakan hanya satu, tetapi tidak dengan hybrid metaheuristik. Sedangkan hyper-heuristic berfokus pada ruang pencarian heuristic (strategi menjadi general) sehingga dapat menangani berbagai contoh masalah dengan karakteristik yang berbeda tanpa memerlukan intervensi ahli. Untuk pembahasan mengenai hyper-heuristic lebih lanjut dapat dilihat pada [10].

Sebagaimana telah disebutkan sebelumnya bahwa salah satu ciri khas dari hyper-heuristic adalah pemisahan letak logika pemrosesan problem domain dan metodologi. Komponen utama pada bagian problem domain pada hyper heuristic adalah fungsi objektif, pembentukan inisial solusi, dan memiliki satu set low-level heuristic. Bagian metodologi memiliki dua tahapan di dalamnya yaitu pemilihan low level heuristic (*heuristic selection*) dan *move acceptance.*

2.2.1 Pembentukan Inisial Solusi

Untuk menghasilkan inisial solusi yang layak, kami mengacu pada metode yang telah umum digunakan sebagai tolak ukur dalam ruang lingkup penelitian exam timetabling problem. Penelitian tersebut dilakukan oleh Carter *dkk* (1996) menggunakan *different heuristic orderings based on graph coloring method* [4].

Proses awal dimulai dengan mendahulukan *exam* yang menghasilkan paling banyak konflik dengan *exam* lainnya. Selanjutnya sebuah *exam* yang tidak memiliki konflik dengan *exam* lainnya yang telah memiliki *timeslot,* akan diletakkan secara random pada suatu *timeslot. Exam* tersebut akan dihapus bersama dengan *exam* yang memiliki konflik jika selama proses tersebut belum ditemukan solusi yang layak. Status *exam* tersebut kembali menjadi *exam* yang belum memiliki *timeslot*. *Exam* yang telah dihapus kemudian ditempatkan pada *timeslot* sama dengan cara sebelumnya. Proses akan berulang hingga solusi yang layak ditemukan.

2.2.2 Metode Pemilihan *Low Level Heuristic*

Salah satu komponen utama pada bagian *problem domain* pada hyper heuristic adalah adanya set low-level heuristic. Hyper-heuristic mengendalikan low-level heuristic dengan tujuan untuk menyediakan sebuah solusi daripada menyediakan solusi secara spesifik untuk problem domain tertentu [11]. Kami mengusulkan 4 low-level heuristics, yaitu:

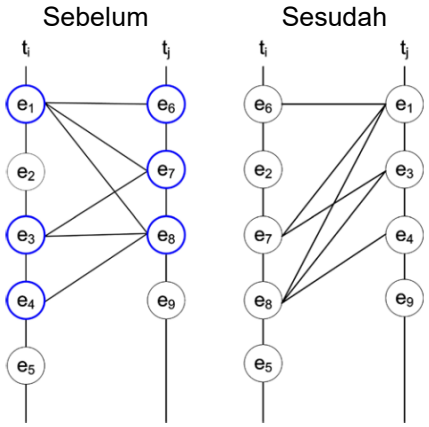
a) Random Move: pilih sebuah *exam* lalu pindahkan *exam* tersebut secara random dari timeslot asal ke timeslot lainnya;
b) Swap Two: secara random, pilih dua *exam*. Kemudian *timeslot* awal milik kedua *exam* tersebut dipindahkan secara acak ke *timeslot* lainnya;

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019

c) Shuffle: Secara random, *timeslot* milik semua *exam* ditukar; dan



Gambar 2. Contoh Heuristik Kempe Chain [3]

d) Swap Kempe Chain: dua set *exam* yang memiliki konflik dipilih dan *timeslot* keduanya ditukar. Contoh dapat dilihat pada Gambar 2. Kondisi awal dapat dilihat bahwa exam e1 dan e3 pada timeslot t_i memiliki peserta ujian yang sama dengan exam e7 pada timeslot t_j. Begitupula dengan exam e1 dan e4 pada timeslot t_i memiliki peserta ujian yang sama dengan exam e8 pada timeslot t_j. Solusi yang dihasilkan dari kedua kondisi tersebut menjadi tidak layak. Solusi yang harus dilakukan yaitu menukar timeslot exam e1, e3 dan e4 menjadi timeslot t_j sedangkan exam e6, e7, dan e8 dipindahkan pada timeslot t_i.

Pada setiap iterasi, keempat set low-level heuristics diatas akan dipilih secara random dengan algoritma *Simple Random* yang telah dikenal sebagai salah satu algoritma yang paling sederhana.

2.2.3 *Move Acceptance*

Proses setelah memilih *low level heuristic* adalah untuk menentukan apakah solusi yang ada akan diterima atau tidak. Untuk mekanisme penerimaan ini, peneliti mengusulkan Hill Climbing dan Simulated Annealing. Untuk selanjutnya *simple random-hill climbing* akan disebut sebagai algoritma 1 pada Gambar 3 dan *simple random-simulated annealing* akan disebut sebagai algoritma 2 pada Gambar 4. Umumnya, pada setiap iterasi, kandidat solusi yang baru terbentuk akan dibandingkan dengan solusi yang telah ada sebelumnya [12].

Umumnya, algoritma dasar dari Hill Climbing adalah selalu menerima solusi yang lebih baik dari solusi sebelumnya. Tetapi pada kedua algoritma, peneliti mengusulkan untuk menambah *delta evaluation* pada tahapan *move acceptance* yaitu dengan menghitung perubahan nilai yang terjadi antara kandidat solusi yang baru terbentuk akan dibandingkan dengan solusi yang telah ada sebelumnya. Menambah penggunaan *delta evaluation* bertujuan untuk mengurangi *time consuming process* [13]. Perbedaannya, pada algoritma 2 terdapat kondisi tambahan lain dimana ketika nilai *delta evaluation* tidak terpenuhi maka fungsi dari *simulated annealing* akan bekerja. Fungsi tersebut dikenal dengan istilah *accepting non improving solution*, yaitu solusi yang lebih buruk dari solusi sebelumnya masih memiliki kemungkinan untuk diterima selama memenuhi probabilitas *simulated annealing* yang dapat dilihat pada persamaan (2).

$$p_t = \exp^{-\left(\frac{deltaEvolution}{T}\right)} \quad (2)$$

Dimana p_t adalah probabilitas pada setiap *move t*, *deltaEvolution* menampung perubahan nilai yang terjadi antara kandidat solusi yang baru terbentuk akan dibandingkan dengan solusi yang telah ada sebelumnya, dan *T* adalah nilai inisial temperatur maksimal [14]. Dikarenakan Framework Hyper-heuristic tidak menyediakan nilai untuk *T* dan temperatur sebagai faktor penurun *α* maka kami menetapkan nilai 1 dan 0.85, untuk masing-masing [15]. Simulated Annealing memiliki inti pada implementasinya yaitu bagaimana temperatur akan mengalami penurunan selama pencarian, faktor

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019

tersebut dikenal sebagai *cooling schedule α.* Iterasi akan berhenti ketika mencapai batas waktu yang telah ditentukan sebelumnya. Adapun pseudocode algoritma yang digunakan dapat dilihat pada gambar 3 dan gambar 4.

```
1   Bentuk InisialSolusi, s(0)
2   Set LowLevelHeuristic
3   Set NilaiInisialFungsiObjektif, f(s)
4   Set TimeLimit
5   Do (!TimeLimit)
6       pilih LowLevelHeuristic secara random
7       LowLevelHeuristic yang terpilih, aplikasikan pada s(0)
8           Get SolusiBaru
9       Hitung delta = f(s) - SolusiBaru
10  IF (delta > 0)
11      f(s) ← SolusiBaru
```

Gambar 3. Algoritma 1

```
1   Bentuk InisialSolusi, s(0)
2   Set LowLevelHeuristic
3   Set NilaiInisialFungsiObjektif, f(s)
4   Set TimeLimit
5   Set InisialTemperatur, T
6   Set DecreasingFactorValue, α
7   Do (!TimeLimit)
8
9       pilih LowLevelHeuristic secara random
10      LowLevelHeuristic yang terpilih, aplikasikan pada s(0)
11          Get SolusiBaru
12      Hitung delta = f(s) – SolusiBaru
13  IF (delta > 0)
14      f(s) ← SolusiBaru
15  ELSE
16
17  IF exp(-(delta/T)) > Random(0,1)
18      f(s) ← SolusiBaru
19  end IF
20  Set T = T * α, after every iteration
21  end while
```

Gambar 4. Algoritma 2

3. Hasil dan Pembahasan

Selain menguji pada data set yang sama, kami juga menggunakan batas waktu eksekusi dan jumlah eksekusi yang sama dengan penelitian yang dilakukan oleh Lei dkk (2015). Pengujian kedua algoritma dilakukan dengan bahasa pemrograman Java di NetBeans IDE 8.0, sistem operasi Windows 10 Pro dengan spesifikasi hardware sebagai berikut: Prosesor *Intel*® Core™ i3-3217U CPU @ 1.80 GHz 1.80 GHz dan *RAM* 8.00 GB. Setiap instansi pada data set dieksekusi sebanyak 10 kali. Batas waktu eksekusi algoritma adalah 1 jam atau setara dengan 3600000 milidetik, sesuai dengan aturan pada hyper-heuristic yang membutuhkan input waktu dalam satuan milidetik dan jumlah eksekusi sebanyak 10 kali.

Performa dua algoritma yang diusulkan merupakan hasil pengujian terhadap 8 instansi yang dipilih secara random (kecuali: PUR93, RYE92, STA83, TRE92, UTE 92). Hasil eksperimen berdasarkan 10 kali eksekusi untuk setiap instansi dapat dilihat pada Tabel 2. Pada Tabel 2, ditampilkan nilai minimal sebagai nilai best dan nilai rata-rata yang dihasilkan selama proses iterasi dibandingkan dengan hasil yang dilaporkan pada literatur.

Walaupun ketiga algoritma tersebut diuji dengan batas waktu yang sama yaitu satu jam dan dieksekusi sebanyak 10 kali, algoritma *simple random – simulated annealing* memperlihatkan

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

performa yang mendominasi hasil terbaik jika dibandingkan dengan *simple random – hill climbing* maupun dari hasil pengujian algoritma dari literatur. Tabel 2 menampilkan *best* dan *avg* sebagai nilai minimum yang diperoleh selama iterasi dan nilai rata-rata. *Bold* menandakan solusi yang lebih baik diantara kedua algoritma lainnya.

Tabel 2. Performa Kedua Algoritma Terhadap 8 Instansi

Problem Instance	Our Results				Lei *dkk* (2015)	
	SR-HC		SR-SA			
	Best	AVG	Best	AVG	Best	AVG
CAR91	5.21	5.5	**4.33**	**4.5**	5.03	5.27
CAR92	4.5	4.62	**4.44**	**4.59**	4.53	4.76
EAR83	**36.0**	38.54	36.49	**37.66**	37.42	38.31
HEC92	11.16	11.6	**11.14**	**11.54**	11.53	11.79
KFU93	**13.83**	**14.66**	14.16	14.81	14.93	15.32
LSE91	11.48	12.12	**11.19**	**11.9**	11.23	11.82
PUR93	-	-	-	-	-	-
RYE92	-	-	-	-	-	-
STA83	-	-	-	-	-	-
TRE92	-	-	-	-	-	-
UTA92	3.64	3.72	**3.61**	**3.69**	3.75	4.01
UTE92	-	-	-	-	-	-
YOR83	39.13	40.2	**38.92**	**40.07**	41.04	41.78

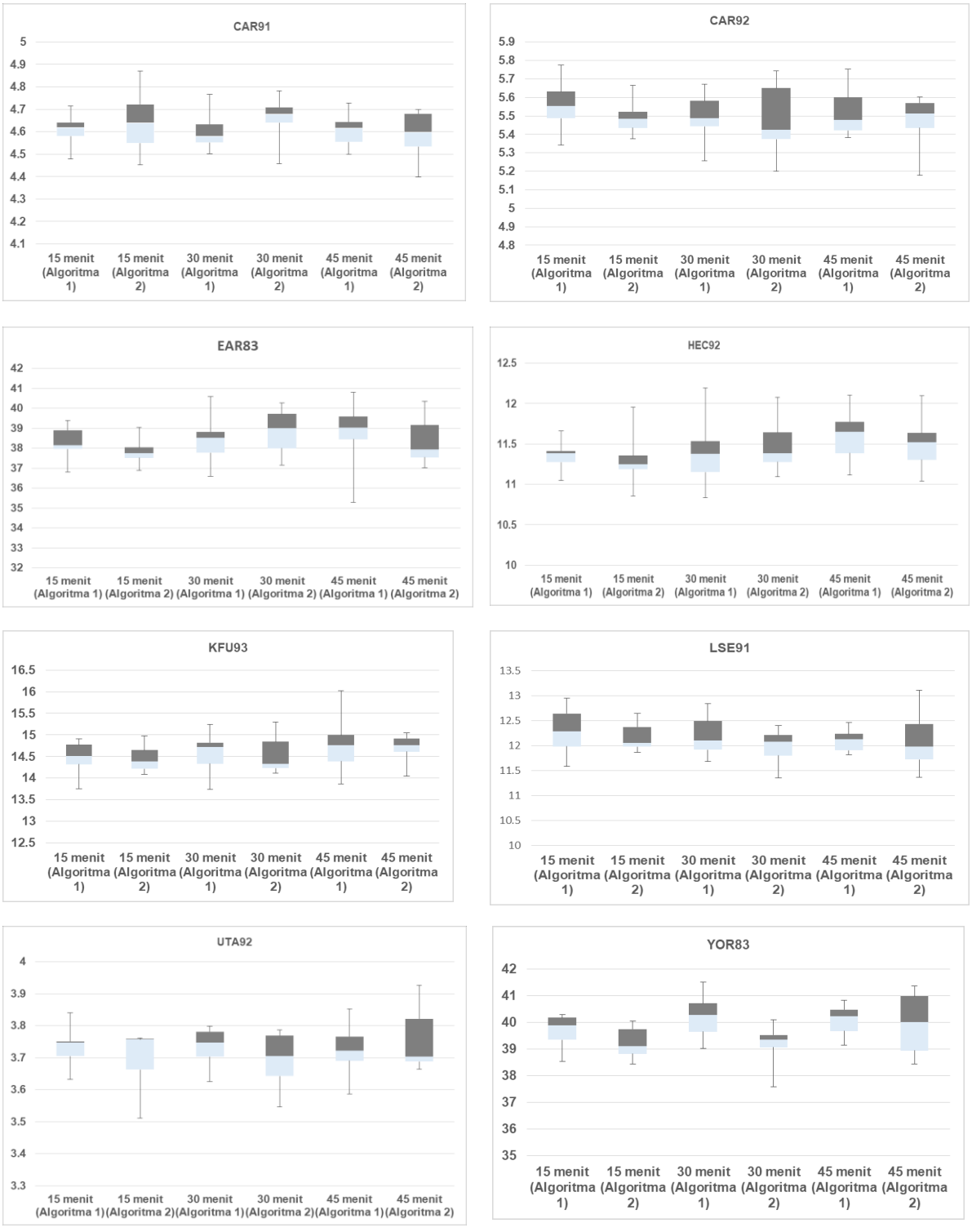
Catatan: SR-HC = Simple Random-Hill Climbing; SR-SA = Simple Random-Simulated Annealing; Best = Nilai Minimum; AVG = Nilai Rata-rata; Solusi lebih baik ditunjukkan dalam *bold*; Tanda "–" merupakan instansi yang tidak diuji.

Selain dibandingkan dengan penelitian Lei dkk (2015), menggunakan waktu berhenti yang sama yaitu selama 1 jam, peneliti juga mencoba untuk melakukan pengujian kedua algoritma dengan waktu kurang dari 1 jam. Mengingat beberapa literatur dengan pengujian terhadap data set ini memerlukan waktu antara 1 hingga 12 jam untuk menemukan hasil yang baik, diantaranya dapat dilihat pada [3, 6,16]. Gambar 5 menampilkan box dan whisker plot data set Carter dari pengujian algoritma 1 dan 2. Setiap box memiliki garis pada kuartil satu (Q1), batas bawah (median-Q1), dan batas atas (Q3-median) serta whisker atas (max-Q3) dan whisker bawah (median-Q1) dari eksekusi sebanyak 10 kali pada tiga variasi waktu (15 menit, 30 menit, dan 45 menit). Garis paling bawah dari box merupakan nilai kuartil satu (Q1), garis di tengah antara box satu dengan lainnya merupakan nilai dari batas bawah, dan garis paling atas dari box merupakan nilai batas atas. Whisker merupakan garis yang berada di atas (akan disebut sebagai "whisker atas") dan di bawah (akan disebut sebagai "whisker bawah"). Dari gambar dapat dilihat bahwa kedua algoritma cenderung menemukan hasil yang lebih baik pada waktu berhenti 45 menit, algoritma 2 (*simple random – simulated annealing*) mendominasi hasil yang lebih baik jika dibandingkan dengan algoritma 1 (*simple random – hill climbing*). Lebih lanjut, hasil pengujian juga ditampilkan pada Tabel 3. Hasil yang ditampilkan merupakan nilai rata-rata dari pengujian selama 10 kali.

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019



Gambar 5. Box dan Whisker Plot dari Algoritma 1 dan 2 pada Data set Carter (Toronto)

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

p-ISSN 2087-1716
e-ISSN 2548-7779

ILKOM Jurnal Ilmiah Volume 11 Nomor 1 April 2019
Terakreditasi peringkat 4 SK. No. 10/E/KPT/2019

Tabel 3. Perbandingan Hasil Pengujian Kedua Algoritma Dengan Variasi Waktu

Problem Instance	15 menit		30 menit		45 menit	
	SR-HC	SR-SA	SR-HC	SR-SA	SR-HC	SR-SA
CAR91	4.60	4.65	4.50	4.65	4.49	**4.39**
CAR92	5.56	5.49	5.50	5.48	5.38	**5.17**
EAR83	38.29	37.82	38.54	38.83	**35.28**	37.02
HEC92	11.33	11.29	11.39	11.63	11.11	**11.04**
KFU93	14.48	14.43	14.67	14.63	**13.86**	14.03
LSE91	12.29	12.18	12.58	12.08	11.82	**11.36**
PUR93	-	-	-	-	-	-
RYE92	-	-	-	-	-	-
STA83	-	-	-	-	-	-
TRE92	-	-	-	-	-	-
UTA92	3.73	**3.70**	3.74	3.77	3.72	3.73
UTE92	-	-	-	-	-	-
YOR83	39.64	39.25	40.51	39.54	39.93	**38.59**

Catatan: SR-HC = Simple Random-Hill Climbing; SR-SA = Simple Random-Simulated Annealing; Solusi lebih baik ditunjukkan dalam *bold*; Tanda "–" merupakan instansi yang tidak diuji.

Dari Tabel 3 dapat dilihat bahwa performa algoritma 2 (*simple random – simulated annealing*) mendominasi dalam menemukan hasil yang lebih baik jika dibandingkan dengan algoritma 1 (*simple random – hill climbing*) dengan waktu berhenti selama 45 menit. Walaupun hasil pada instansi UTA92 menunjukkan bahwa hasil yang baik dapat ditemukan pada waktu berhenti selama 15 menit, tetapi perbedaan nilai dengan hasil yang lainnya menunjukkan rentang yang tidak terlalu signifikan.

Eksperimen terakhir pada penelitian ini adalah dengan membandingkan hasil pada Tabel 3 dengan penelitian yang dilakukan oleh Carter *dkk* (1996) sebagai penelitian awal yang menggunakan data set Carter (Toronto). Perbandingan dapat dilihat pada Tabel 4.

Dari Tabel 4 dapat dilihat bahwa performa kedua algoritma mendominasi dalam menemukan hasil yang lebih baik pada waktu berhenti selama 45 menit jika dibandingkan dengan penelitian yang dilakukan oleh Carter *dkk* (1996) pada waktu berhenti selama 12 jam, atau setara dengan 720 menit.

4. Kesimpulan dan Saran

Penelitian ini melakukan pengujian dua pasang algoritma terhadap salah satu data set yang biasa digunakan oleh para peneliti yaitu data set Carter (Toronto) melalui pendekatan hyperheuristic. Beberapa penelitian yang mendiskusikan data set Carter mencantumkan waktu komputasi, tetapi beberapa ada yang tidak mencantumkan. Oleh karena itu, peneliti memilih salah satu literatur, yaitu oleh Lei *dkk* (2015) yang mencantumkan nilai batas waktu eksekusi dan jumlah eksekusi, untuk dijadikan pembanding. Adapun nilai batas waktu eksekusi dan jumlah eksekusi pada penelitian ini mengikuti literatur tersebut. Walaupun pengujian dilakukan dengan batas waktu yang sama yaitu satu jam dan dieksekusi sebanyak 10 kali, algoritma kedua (*simple random – simulated annealing*) memperlihatkan performa yang mendominasi hasil terbaik jika dibandingkan dengan algoritma 1 (*simple random – hill climbing*) maupun hasil pengujian algoritma pada literatur.

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

Tabel 4. Perbandingan Hasil Pengujian kedua Algoritma pada Variasi Waktu dengan Carter dkk (1996)

Problem Instance	15 menit		30 menit		45 menit		12 jam
	SR-HC	SR-SA	SR-HC	SR-SA	SR-HC	SR-SA	Carter *dkk* (1996)
CAR91	4.60	4.65	4.50	4.65	4.49	**4.39**	7.10
CAR92	5.56	5.49	5.50	5.48	5.38	**5.17**	6.20
EAR83	38.29	37.82	38.54	38.83	**35.28**	37.02	36.40
HEC92	11.33	11.29	11.39	11.63	11.11	11.04	**10.80**
KFU93	14.48	14.43	14.67	14.63	**13.86**	14.03	14.00
LSE91	12.29	12.18	12.58	12.08	11.82	11.36	**10.50**
PUR93	-	-	-	-	-	-	-
RYE92	-	-	-	-	-	-	-
STA83	-	-	-	-	-	-	-
TRE92	-	-	-	-	-	-	-
UTA92	3.73	3.70	3.74	3.77	3.72	3.73	**3.50**
UTE92	-	-	-	-	-	-	-
YOR83	39.64	39.25	40.51	39.54	39.93	**38.59**	41.70

Catatan: SR-HC = Simple Random-Hill Climbing; SR-SA = Simple Random-Simulated Annealing; Solusi lebih baik ditunjukkan dalam *bold*; Tanda "–" merupakan instansi yang tidak diuji.

Peneliti juga melakukan eksperimen pengujian algoritma pada waktu yang bervariasi tetapi kurang dari 1 jam (15 menit, 30 menit, dan 45 menit) yang kemudian dibandingkan dengan pengujian yang dilakukan oleh Carter *dkk* dalam waktu 12 jam. Dari hasil pengujian dapat dilihat bahwa, semakin kecil waktu berhenti tidak berarti bahwa hasil yang diperoleh akan lebih baik. Pada penelitian ini dapat disimpulkan bahwa daripada menggunakan waktu berhenti selama satu jam, pengujian dapat dilakukan dengan waktu berhenti 45 menit per instansi. Waktu tersebut lebih cepat jika dibandingkan dengan waktu berhenti yang biasa digunakan oleh penelitian sebelumnya, 1 hingga 12 jam, untuk menemukan hasil terbaik.

Penelitian kali ini untuk melihat bagaimana performa dari dua pasang algoritma yang diusulkan melalui pendekatan hyperheuristic dan hanya diujikan pada 8 instansi dalam waktu berhenti yang bervariasi. Sehingga untuk penelitian selanjutnya, peneliti bertujuan untuk melakukan pengujian pada seluruh instansi, menginvestigasi performa dari algoritma yang diusulkan pada data set *exam timetabling* lainnya, seperti ITC 2007, serta menguji algoritma baru pada data set Carter (Toronto). Selain itu, untuk memastikan tingkat generalisasi dari hyperheuristic maka algoritma yang diusulkan dapat diuji pada *problem domain* yang berbeda.

Daftar Pustaka

[1] E. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined local search approach to exam timetabling problems," *IIE Trans.*, vol. 36, no. 6, pp. 509–528, 2004.

[2] C. Weng, H. Asmuni, B. Mccollum, and P. Mcmullan, "A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems," *Inf. Sci. (Ny).*, vol. 283, pp. 1–21, 2014.

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44

[3] P. Demeester, B. Bilgin, P. De Causmaecker, and G. Vanden, "A hyperheuristic approach to examination timetabling problems : benchmarks and a new problem from practice," pp. 83–103, 2012.
[4] B. Bilgin, E. Ozcan, and E. E. Korkmaz, "An Experimental Study on Hyper-heuristics and Exam Timetabling," in *Practice and Theory of Automated Timetabling VI*, E. Burke and H. Rudová, Eds. 2007, pp. 394–412.
[5] M. Kalender, A. Kheiri, E. Özcan, and E. K. Burke, "A greedy gradient-simulated annealing selection hyper-heuristic," *Soft Comput.*, vol. 17, no. 12, pp. 2279–2292, 2013.
[6] Y. Lei, M. Gong, L. Jiao, and Z. Yi, "A memetic algorithm based on hyper-heuristics for examination timetabling problems," *Int. J. Intell. Comput. Cybern.*, vol. 8, no. 2, pp. 139–151, 2015.
[7] A. Muklason, A. J. Parkes, E. Özcan, B. McCollumb, and P. McMullan, "Fairness in examination timetabling: Student preferences and extended formulations," *Appl. Soft Comput.*, vol. 55, pp. 302–318, 2017.
[8] E. K. Burke, "Hybrid variable neighbourhood hyper-heuristics for exam timetabling problems Hybrid Variable Neighborhood HyperHeuristics for Exam Timetabling Problems," no. July, 2017.
[9] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A Reinforcement Learning - Great-Deluge Hyper-Heuristic for Examination Timetabling," *Int. J. Appl. Metaheuristic Comput.*, vol. 1, no. 1, pp. 39–59, 2010.
[10] E. K. Burke, T. Curtois, M. Hyde, G. Ochoa, and J. a Vazquez-Rodriguez, "HyFlex: A Benchmark Framework for Cross-domain Heuristic Search," *Arxiv Prepr. arXiv11075462*, vol. abs/1107.5, p. 28, 2011.
[11] Y. Lei, M. Gong, L. Jiao, and Y. Zuo, "A memetic algorithm based on hyper-heuristics for examination timetabling problems," *Int. J. Intell. Comput. Cybern.*, vol. 8, no. 2, pp. 139–151, 2015.
[12] E. Özcan, Y. Bykov, M. Birben, and E. K. Burke, "Examination Timetabling Using Late Acceptance Hyper-heuristics," University of Nottingham, 2009.
[13] P. Ross, D. Corne, and H. Fang, "Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation."
[14] R. Bai and G. Kendall, "An Investigation Of Automated Planograms Using A Simulated Annealing Based Hyper-Heuristic," no. June 2015, pp. 0–22, 2005.
[15] E. Soubeiga, "Development and Application of Hyper Huristics to Personnel Scheduling," 2003.
[16] M. W. Carter, G. Laporte, and S. Y. Lee, "Examination Timetabling: Algorithmic Strategies and Applications," *J. Oper. Res. Soc.*, pp. 373–383, 1996.

DOI: http://dx.doi.org/10.33096/ilkom.v11i1.420.34-44
