Solver Penjadwal Ujian Otomatis Dengan Algoritma *Maximal Clique* dan Hyper-heuristics

Ahmad Muklason
Departemen Sistem Informasi, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Jl. Raya ITS, Kampus ITS Sukolilo, Surabaya, 60111
e-mail: mukhlason@is.its.ac.id

Abstrak

Permasalahan penjadwalan ujian adalah permasalahan berulang yang terjadi setidaknya satu kali dalam satu semester di lingkungan akademik, baik sekolah maupun perguruan tinggi. Menentukan jadwal dengan memastikan tidak ada satupun mahasiswa yang harus menempuh ujian dua mata kuliah di waktu yang sama, penentuan ruang ujian, dan penjadwalan pengawas ujian adalah pekerjaan yang sangat menyita waktu. Karena itu solver penjadwal ujian otomatis sangat diperlukan. Lebih dari itu, secara teoritis, permasalahan optimasi penjadwalan ujian ini merupakan NP-complete problem, dimana belum ada algoritma eksak yang mampu menyelesaiakan permasalahan ini dalam waktu non-polinomial. Sehingga permasalahan ini banyak menarik perhatian para peneliti, khususnya di bidang riset operasi dan kecerdasan buatan selama puluhan tahun belakangan ini. State-of-the-art pendekatan untuk memecahkan permasalahan ini adalah metode heuristic sekuensial berdasarkan permalahan pewarnaan graf dan meta-heuristic. Makalah ini membahas usulan metode baru yaitu metode heuristic sekuensial berdasarkan konsep maximal clique pada teori graf digabung dengan metode hyper-heuristic. Hasil penelitian komputasi menunjukkan bahwa metode ini sangat efektif untuk memecahkan permasalahan penjadwalan ujian dan lebih unggul jika dibandingkan dengan hasil penelitian sebelumnya dengan metode yang lain.

Kata kunci: *permasalahan penjadwalan ujian, hyper-heuristic, maximal clique, penjadwalan otomatis, meta-heuristic.*

Abstract

The examination timetabling problems is repetitive work at least once in each semester within academic institutions either schools or universities. Creating a timetable that guarantees no student has to sit for two exams in the same time with room assignment and invigilators scheduling is tedious work. Therefore, a solver for automated examination timetable is urgently needed. In addition, theoretically examination timetabling problem optimization is categorized as NP-complete problem, in which there is no exact algorithm successfully invented to solve the problem. For decades, this difficult problem has attracted many researchers especially in the area of operation research and artificial intelligence. The state-of-the art approach to solving this problem is sequential heuristics based on graph coloring problem and meta-heuristics. This paper proposes a new method using sequential heuristic based on maximal clique concept in the graph theory combined with hyper-heuristic approach. The computational experiment results show that the proposed method is very effective and competitive compared to prior results reported in the literature with different methods.

Keywords*: examination timetabling problem, hyper-heuristics, maximal clique, automated timetabling, meta-heuristics*

1. Pendahuluan

Penjadwalan adalah salah satu kajian menarik dalam bidang permasalahan optimisai kombinatorik. Secara umum, permasalahan optimasi kombinatorik adalah kajian matematis untuk mencari solusi optimal pada penyusunan, pengelompokan, pengurutan, atau pemilihan objek diskrit yang biasanya jumlahnya terbatas (*finite number*) [1]. Penjadwalan kelas dan ujian serta penjadwalan personel adalah permasalahan penjadwalan yang telah banyak dikaji di literatur. Makalah ini, fokus membahas pada permasalahan penjadwalan ujian.

Secara lebih khusus, pada [2] permasalahan penjadwalan didefinisikan sebagai permasalahan dengan empat parameter, yaitu *T, S, P, B* yang secara berurutan mewakili himpunan terbatas (*finite set*) dari *timeslot*, sumber daya (misal: ruangan), pertemuan, dan batasan. Permasalahan ini menyelesaikan pengalokasian *timeslot* dan sumber daya pada pertemuan dengan memenuhi batasan semaksimal mungkin.

Lebih khusus lagi, pada [3] penjadwalan ujian dapat dirumuskan sebagai *3-tuple <U, T, B>*, dimana $U=\{u_1, u_2, ..., u_N\}$ adalah himpunan dari ujian (mata kuliah atau mata pelajaran yang diujikan), $T=\{t_1, t_2, ..., t_M\}$ adalah *ordered list* dari *timeslot*, dan $B=\{b_1, b_2, ..., b_K\}$ adalah himpunan batasan. Sehingga permasalahan penjadwalan ujian dapat didefinisakan sebagai

permasalahan pencarian pengalokasian terbaik untuk $\forall_x, \exists_y (u_x = t_y)$ yang berarti ujian u_x dijadwalkan pada *timeslot* t_y dimana $u_x \in U$ dan $t_y \in T$, sedemikian hingga semua batasan *B* terpenuhi.

Dari sudut pandang teori kompleksitas komputasi, permasalahan penjadwalan ini telah dibuktikan di [4] sebagai *NP-Complete problem*. Dengan demikian belum ada algoritma eksak yang mampu menemukan solusi optimal dalam waktu polinomial. Sehingga permasalahan ini sangat menarik untuk terus dikaji. Di literatur, telah ada beberapa *benchmark dataset* untuk permasalahan penjadwalan ujian ini. Diantaranya yang sudah dikaji secara luas adalah Carter[5] dan ITC 2007[6] *dataset*. Sementara benchmark yang terbaru dapat dilihat di[7]. Dalam makalah ini digunakan Carter dataset.

Permasalahan penjadwalan ujian pada Carter dataset bertujuan untuk meminimalkan *proximity cost, P,* yang secara ringkas dapat diformulasikan pada persamaan (1) – (3):

$$P = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{i,j} \; W_{|t_i - t_j|}}{S} \quad (1)$$

Dimana,

$$W_{|t_i - t_j|} = 2^{5-|t_j - t_i|} \; jika \; 1 \leq |t_j - t_i| \leq 5, 0 \; jika \; tidak \quad (2)$$

Sedemikian hingga, hard constrain berikut terpenuhi:

$$\forall_{i \neq j}, t_i \neq t_j \; jika \; C_{i,j} > 0 \quad (3)$$

Pada pada persamaan (1) – (3), *N* adalah total jumlah mahasiswa, $C_{i,j}$ adalah jumlah mahasiswa yang mengambil ujian untuk mata kuliah *i* dan *j*, $W_{|t_i - t_j|}$ adalah bobot penalty ketika ujian *i* dan *j* masing-masing dijadwalkan pada *timeslot* t_i dan t_j. Selanjutnya persamaan (1) bisa disebut sebagai *soft constraints* dan persamaan (2) disebut sebagai *hard constraints*.

Carter dataset terdiri dari 13 *problem instance* yang berasal dari masalah penjadwalan ujian dari berbagai universitas. Karakteristik dari ketigabelas *problem instance* ini diringkas pada Tabel 1. Di literatur, sudah banyak penelitian sebelumnya untuk menyelesaikan permasalahan penjadwalan ujian dengan Carter dataset ini. *State-of-the-art* metode yang digunakan untuk menghasilkan solusi yang *feasibel* (memenuhi semua *hard constraints*) adalah *sequential heuristic* dimana permasalahan penjadwalan ujian dimodelkan sebagai permasalahan pewarnaan graf. Dalam model pewarnaan graf, ujian direpresentasikan oleh *node*, dua ujian dengan setidaknya ada satu mahasiswa yang sama direpresentasikan sebagai *edge* yang menghubungkan dua *node*, dan *timeslot* direpresentasikan oleh warna node. Dengan model ini, permasalahan pengalokasian *timeslot* ke ujian menjadi permasalahan pewarnaan *node* dimana setiap *node* yang terhubung *edge* harus diberi warna yang berbeda. Sementara itu *state-of-the-art* algoritma untuk optimasi solusi *feasible* (meminimalkan pelanggaran *soft constraints*) adalah algoritma *meta-heuristics*.

State-of-the-art algoritma untuk menyelesaikan permasalahan penjadwalan ujian bisa dilihat di[8]. Algoritma yang dilaporkan di literatur terbaru sebagai metode yang efektif untuk menyelesaikan penjadwalan ujian ini didominasi oleh algoritma *meta-heuristics*. Diantaranya adalah: *simulated annealing*[9], *great deluge*[10], *tabu search*[11], *large neighborhood search*[12], *variable neighborhood search*[13], *iterated local search*[14], *greedy randomized adaptive search procedures (GRASP)*[15], *memetic algorithm*[16], *intelligent water drop algorithm*[17], *genetic algorithm*[18], *ant algorithm*[19], *artificial immune algorithm*[20], *harmony search*[21], *evolutionary ruin and stochastic rebuild*[22], *fish swarm*[23], *honey-bee mating*[24], *artificial bee colony*[25], *imperialist swarm-based optimization*[26], dan *particle swarm optimization algorithms*[27].

Tabel 1 *Carter Dataset*

Problem Instances	No.Of Exams	No.Of Students	Conflict Density	No.Of Timeslots
CAR91	682	16925	0.13	35
CAR92	543	18419	0.14	32
EAR83	190	1125	0.27	24
HEC92	81	2823	0.42	18
KFU93	461	5349	0.06	20
LSE91	381	2726	0.06	18
PUR93	2419	30029	0.03	42
RYE92	486	11483	0.07	23
STA83	139	611	0.14	13
TRE92	261	4360	0.18	23
UTA92	622	21266	0.13	35
UTE92	184	2749	0.08	10
YOR83	181	941	0.29	21

Umumnya, kelemahan dari algoritma *meta-heuristic* adalah diperlukan *parameter tuning* yang intensif dan memerlukan pengetahuan *problem domain* yang spesifik. Sehingga, untuk *problem instance* yang berbeda diperlukan parameter tuning yang berbeda juga. Jika tidak, performa algoritma akan bagus pada suatu *problem instance* tetapi sangat buruk pada *problem instance* yang lain. Oleh karena itu, untuk mengatasi permasalahan ini ide *hyper-heuristic* muncul.

Secara umum, di[28] *hyper-heuristic* dapat didefinisikan sebagai kumpulan pendekatan dengan tujuan untuk otomasi proses, biasanya menggunakan *machine learning,* untuk: (1) memilih dan mengkombinasikan heuristics yang lebih sederhana, atau (2) menghasilkan heuristic baru dengan komponen heuristic yang sudah ada, untuk memecahkan permasalahan pencarian komputasi yang sangat sulit dilakukan secara manual (*hard computational search problem*). Gambar 1 mengilustrasikan *framework* dari hyper-heuristic sebagaimana dijelaskan di[29].

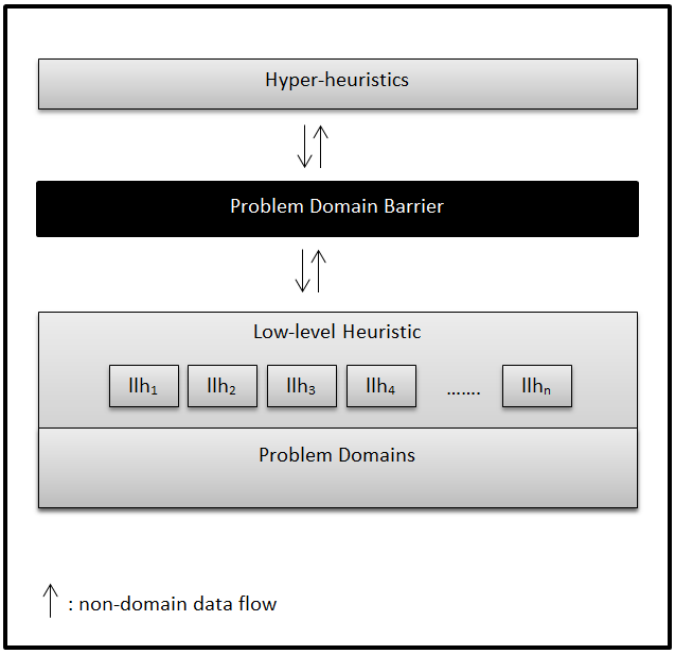
Sebagaimana diilustrasikan pada Gambar 1, pada *hyper-heuristic* ada *problem domain barrier* yang mengakibatkan strategi *hyper-heuristic* tidak bersinggungan langsung dengan *solution space* (sebagaimana metode meta-heuristics pada umumnya), tetapi hanya bersinggungan dengan *lower-level heuristics*. Dengan bahasa lain, algoritma pencarian *hyper-heuristic* bekerja diatas *search space* berupa *lower-level heuristics* sementara *meta-heuristic* bekerja diatas *search space* berupa *solution space*. Dengan *search space* yang lebih tinggi ini, metode *hyper-heuristic* tidak tergantung pada *problem domain* tertentu saja, sehingga tidak diperlukan *parameter tuning* untuk setiap problem domain secara manual. Mekanisme *hyper-heuristic* mampu melakukan otomasi proses *parameter tuning* ini.

Dibanding *meta-heuristic*, *hyper-heuristic* masih sangat jarang diteliti untuk memecahkan permasalahan penjadwalan ujian. Diantara penelitian sebelumnya dengan *hyper-heuristic* adalah[30], yang membandingkan beberapa strategi *move acceptantance* dan *low-level heuristic selection learning*. Pada makalah ini akan digunakan metode *maximal clique*, karena terbukti efektif digunakan pada permasalahan pewarnaan graf, untuk inisiasi solusi *feasible* dan diusulkan strategi hyper-heuristic baru untuk optimasi sebagaimana akan dibahas pada bagian 2.

2. Metode

Secara umum algoritma yang diusulkan pada makalah ini terdiri dari dua fase. Fase 1 bertujuan untuk mengkonstruksi solusi inisial yang feasible, yaitu memenuhi semua *hard constraints*. Sedangkan fase 2 bertujuan untuk mengoptimalkan solusi inisial, yaitu meminimumkan jumlah penalti akibat pelanggaran terhadap *soft constraints*. Pada fase 1 digunakan algoritma *sequential heuristic* berdasarkan *maximal clique*. Dalam teori graf, *clique* adalah sub-graf komplit dimana setiap pasang node dihubungkan oleh *edge*. Dalam konteks ukuran *maximal clique* merepresentasikan jumlah minimum *timeslots* yang dibutuhkan untuk membuat jadwal yang *feasibel*.

ISSN (Printed) : 2579-7271
ISSN (Online) : 2579-5406



Gambar 1 *Hyper-heuristic Framework*

Algoritma untuk mengkonstruksi jadwal inisial secara detail dapat dilihat pada Algoritma 1. Pertama, dicari *maximal clique* dari graf yang merepresentasikan permasalahan penjadwalan ujian. Kemudian, setiap *node* atau *vertex* pada *maximal clique* dialokasikan pada warna yang berbeda yang berarti setiap mata ujian pada *maximal clique* dialokasikan timeslot yang berbeda. Kedua, sisa dari *vertex* pada graf yang belum diwarnai diurutkan berdasarkan *saturation degree*, yaitu jumlah warna yang masih feasibel selama proses pengalokasian warna. *Vertex* dengan *saturation degree* paling kecil harus diurutkan paling awal, dan seterusnya. Kemudian, berdasarkan urutan ini, setiap vertex dialokasikan pada warna yang feasibel. Proses ini diulang sampai semua vertex berhasil diwarnai.

Sedangkan algoritma *hyper-heuristics* yang digunakan pada fase 2 secara detail dijelaskan pada Algoritma 2. Secara garis besar algoritma *hyper-heuristic* yang ditunjukkan oleh Algoritma 2 terdiri dari dua strategi. Strategi pertama ditujukan untuk memilih *low-level heuristic* (lihat Tabel 2) di setiap iterasinya, yaitu dengan menggunakan *self adaptive learning* yang diadaptasi dari[29]. Strategi yang kedua adalah *move acceptance* yang menentukan apakah solusi baru yang dihasilkan pada setiap iterasi diterima atau tidak untuk menggantikan solusi dari iterasi sebelumnya. Dalam Algoritma 2, *great deluge (GD)*[30] diadaptasi.

Selain *self adaptive learning (SA)*, dalam penelitian ini *simple random selection* (SR) untuk pemilihan *low-level heuristic* juga dikaji. Demikian juga untuk *move acceptance*, dua alternatif lain yaitu *late acceptance (LA)*[31] dan *hill climbing (HC)* juga dikaji. Sehingga menghasilkan 6 kombinasi strategi *hyper-heuristic* yang berbeda. Kebaruan dari algoritma yang diusulkan pada makalah ini adalah penggabungan *maximal clique* dengan *saturation degree* pada *sequential heuristic* untuk konstruksi solusi inisial yang *feasible* dan penggabungan *self adaptive learning* dengan *great deluge* dalam kerangka kerja *hyper-heuristic* untuk optimasi solusi inisial.

```
1: procedure INITIATESOLUTION(id)
2:     Generate a Graph G from the problem instance.
3:     Find maximal clique MC from G.
4:     //Colour starts from 1
5:     Assign a differing colour number to each vertex, v ∈ MC.
6:     L ← list of remaining uncoloured vertices.
7:     repeat
8:         Order vertices in L by non-increasing saturation degree.
9:         Assign the first vertex, f, from L with a feasible lowest number colour.
10:        Remove f from L.
11:        Update the saturation degree of vertices connected to f.
12:    until All vertices are assigned to colour
13:    store the coloured graph G in solution memory index i.
14: end procedure
```

Algoritma 1 *Maximal Clique based Sequential heuristic*

3. Hasil dan Pembahasan

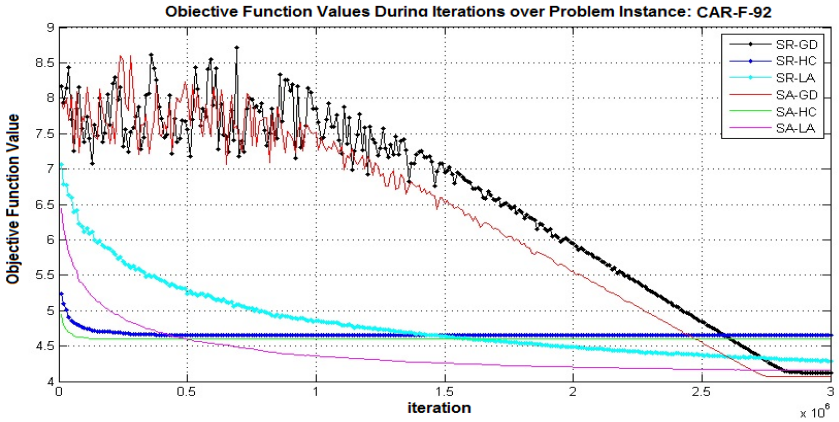
Secara garis besar, ada dua jenis eksperimen yang dilakukan dalam penelitian ini. Pertama eksperimen untuk mengkonstruksi solusi inisial dan yang kedua untuk optimasi solusi inisial. Kedua eksperimaen dilakukan pada semua ketiga belas *problem instance* pada Tabel 1.

3.1. Hasil Eksperimen Mengkonstruksi Solusi Inisial yang *Feasible*

Untuk masing-masing *problem instance*, Algoritma 1 dijalankan 10 kali. Hasil eksperimen menunjukkan bahwa Algoritma 1 selalu berhasil menghasilkan solusi yang *feasible* untuk semua *problem instance* dalam waktu kurang dari 1 menit. Ini menunjukkan bahwa *sequential heuristic* berdasarkan gabungan *maximal clique* dan *saturation degree* sangat efektif untuk mengkonstruksi solusi yang *feasible*.

3.2. Hasil Eksperimen Optimasi Solusi Inisial Dengan Hyper-heuristics

Perbandingan performa algoritma dari enam kombinasi yang berbeda ini ditampilan oleh Gambar 2. Dari Gambar 2, terlihat bahwa algoritma SA-GD paling unggul dibanding kombinasi algoritma yang lain. Hal ini disebabkan SA-GD mampu keluar dari lokal optima dengan cara dapat menerima solusi baru meskipun lebih jelek dari solusi dari iterasi sebelumnya.



Gambar 2 Perbandingan Performa Algoritma Hyper-heuristic

Tabel 2 *Low-level Heuristics (LLH)*

LLH	Nama	Keterangan
LLH_1	Random Move	Satu ujian dipilih secara acak, lalu timeslotnya diganti secara acak.
LLH_2	Random Move Two	Dua ujian dipilih secara acak, lalu timeslotnya diganti secara acak.
LLH_3	Swap	Dua ujian dipilih secara acak, lalu timeslotnya ditukar.
LLH_4	Kempe Chain	Dua himpunan ujian yang tidak saling konflik satu sama lain, timeslotnya ditukar.

Sementara itu, perbandingan hasil dari algoritma yang kita usulkan dengan hasil di literatur dapat dilihat di Tabel 3. Dari Tabel 3 dapat dilihat bahwa algoritma yang diusulkan dalam makalah ini lebih unggul dari hasil dari penelitian lain yang dilaporkan di literatur. Terlihat dari 13 *problem instance,* algoritma SA-GD lebih unggul pada 8 problem instance. Hasil ini juga membuktikan bahwa dibanding algoritma *meta-heuristics* algoritma *hyper-heuristic* yang kita usulkan lebih generic, karena tanpa *parameter tuning* khusus, algoritma yang kita usulkan bisa lebih unggul pada jauh lebih banyak *problem instance*.

```
1: procedure SOLVE(ProblemDomain P)
2:     Set stopping condition St
3:     Set M ← P.getNumberOfHeuristics()
4:     Set a list of low-level heuristic indices, LH of size S
5:     Set a list of Well-performed low-level heuristic indices, WH
6:     // Fill LH with the index of low-level heuristics predefined in P chosen randomly
7:     for i=0 to S-1 do
8:        LH.add(getRand(M))
9:     end for
10:    P.initiateSolution(0)
11:    S_best ← P.getFunctionValue(0)
12:    S_curr ← P.getFunctionValue(0)
13:    B ← P.getFunctionValue(0)
14:    Set decay rate α
15:    while St is not met do
16:       if LH is not empty then
17:          // get the earliest element of LH and delete it from LH
18:          l ← LH.poll()
19:       else
20:          fill 75% elements of LH with elements from WH
21:          fill 25% elements of LH with the index of low-level heuristics predefined in P chosen randomly.
22:          Clear the element of WH
23:          l ← LH.poll()
24:       end if
25:       newFunctionVal ← P.applyLLH(l,0,1)
26:       if newFunctionVal≤ S_curr OR newFunctionVal ≤ B then
27:          //the new solution is accepted
28:          P.copySolution(1,0)
29:          S_curr ← newFunctionVal
30:          WH.add(l)
31:          if newFunctionVal<S_best then
32:             S_best ← newFunctionVal
33:          end if
34:       end if
35:       B ← B-α
36:    end while
37:    return S_best
38: end procedure
```

Algoritma 2 *Self-adaptive Great Deluge Algorithm*

4. Kesimpulan

Kontribusi utama dari makalah ini adalah algoritma baru dalam framework *hyper-heuristic* yang merupakan gabungan (*hybrid*) dari beberapa algoritma, yaitu: *maximal clique – saturation degree based sequential heuristic* dan *self adaptive learning – great deluge hyper-heuristic*. Hasil penelitian komputasi menunjukkan bahwa algoritma ini sangat efektif dan kompetitif bahkan lebih unggul dengan metode lainya yg dilaporkan di literatur. Kelanjutan dari penelitian ini kedepanya adalah uji coba algoritma pada kasus nyata dilapangan yang lebih kompleks, i.e. melibatkan penjadwalan ruang dan pengawas ujian. Selain itu, perbaikan performa algoritma juga menarik untuk diinvestigasi.

Tabel 3 Perbandingan Hasil dari Algoritma yang diusulkan Dengan hasil Di literature

Instance	Our Approach		BENCHMARK (BEST)				
	SA-GD	SA-LA	(Burke,2012) [32]	(Sabar,2012) [33]	(Rahman,2014) [34]	(Burke,2014) [35]	(Abdullah,2013) [36]
CAR-F-92-I	**3.93**	4.00	4.22	4.7	4.41	4.31	3.94
CAR-S-91-I	4.78	4.86	5.03	5.14	5.12	5.19	**4.76**
EAR-F-83-I	33.03	**32.97**	36.06	37.86	36.91	35.79	33.61
HEC-S-92-I	10.39	**10.31**	11.71	11.9	11.31	11.19	10.56
KFU-S-93	**13.24**	13.37	16.02	15.3	14.75	14.51	13.44
LSE-F-91	**10.07**	10.33	11.15	12.33	11.41	10.92	10.87
PUR-S-93-I	**4.71**	5.21	NA	5.37	5.87	NA	NA
RYE-F-92	**8.27**	8.27	9.42	10.71	9.61	NA	8.81
STA-F-83-I	**157.08**	157.13	158.86	160.12	157.52	157.18	157.09
TRE-S-92	8.08	8.05	8.37	8.32	8.76	8.49	**7.94**
UTA-S-92-I	3.28	3.33	3.37	3.88	3.54	3.44	**3.27**
UTE-S-92	**24.81**	24.85	27.99	32.67	26.25	26.7	25.36
YOR-F-83-I	**35.35**	35.79	39.53	40.53	39.67	39.47	35.74

Daftar Pustaka

[1] Eugene L. Lawler. Combinatorial Optimization: Networks and Matroids. Holt Rinehart and Winston, 1976.
[2] Edmund K. Burke, Graham Kendall, Mustafa Misir, Ender Ozcan, EK Burke, G Kendall, and M Misir. Applications to timetabling. In Handbook of Graph Theory, chapter 5-6, pages 445–474, 2004.
[3] Edmund K. Burke, Graham Kendall, Mustafa Misir, and Ender Ozcan. Monte Carlo hyper-heuristics for examination timetabling. Annals of Operations Re- search, 196(1): 73–90, 2012.
[4] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In Edmund Burke and Peter Ross, editors, Practice and Theory of Automated Timetabling, volume 1153 of Lecture Notes in Computer Science, pages 281–295. Springer Berlin Heidelberg, 1996.
[5] Michael W. Carter, Gilbert Laporte, and Sau Y. Lee. Examination timetabling: Algorithmic strategies and applications. Journal of the Operational Research Society, 47(3): 373–383, 1996.
[6] Barry McCollum, Paul McMullan, Andrew J. Parkes, Edmund K. Burke, and Rong Qu. A new model for automated examination timetabling. Annals of Operations Research, 194(1): 291–315, 2012.
[7] Muklason A., Parkes A.J., McCollum B. and Ozcan E. Fairness in Examination Timetabling Problems, Journal of Applied Soft Computing, 55:302-318, 2017. DOI:10.1016/j.asoc.2017.01.026
[8] Rong Qu, Edmund K. Burke, Barry McCollum, Liam T.G. Merlot, and Sau Y. Lee. A survey of search methodologies and automated system development for examination timetabling. Journal of Scheduling, 12(1): 55–89, 2009.
[9] Jonathan M. Thompson and Kathryn A. Dowsland. General cooling schedules for a simulated annealing based timetabling system. In Edmund Burke and Peter Ross, editors, Practice and Theory of Automated Timetabling, volume 1153 of Lecture Notes in Computer Science, pages 345–363. Springer Berlin Heidelberg, 1996.
[10] Edmund K. Burke, Yuri Bykov, James Newall, and Sanja Petrovic. A time- predefined local search approach to exam timetabling problems. IIE Transactions, 36(6):509–528, 2004.
[11] A. Hertz. Tabu search for large scale timetabling problems. European Journal of Operational Research, 54(1):39–47, 1991.
[12] Salwani Abdullah, Samad Ahmadi, Edmund K. Burke, and Moshe Dror. In- vestigating Ahuja-Orlin's large neighbourhood search approach for examination timetabling. OR Spectrum, 29(2):351–372, 2007.
[13] Samad Ahmadi, Rossano Barone, Peter Cheng, Edmund K. Burke, Peter Cowl- ing, and Barry McCollum. Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. In multidisciplinary inter- national scheduling: theory and applications (MISTA 2003), pages 155–171, 2003.
[14] Edmund K. Burke, Tim Curtois, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Sanja Petrovic, Jos´e A. Va´zquez-Rodr´ıguez, and Michel Gendreau. Iter- ated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In IEEE congress on evolutionary computation, pages 1–8. IEEE, 2010.
[15] S. Casey and J. Thompson. GRASPing the examination scheduling problem. In E. Burke and P. DeCausmaecker, editors, Practice and Theory of Automated Timetabling IV, volume 2740 of Lecture Notes in Computer Science, pages 232– 244, 2003.
[16] Edmund K. Burke, James P. Newall, and Rupert F. Weare. A memetic algo- rithm for university exam timetabling. In Edmund Burke and Peter Ross, editors, Practice and Theory of Automated Timetabling, volume 1153 of Lecture Notes in Computer Science, pages 241–250. Springer Berlin Heidelberg, 1996.
[17] Bashar A. Aldeeb, Norita Md Norwawi, Mohammed A. Al-Betar, and Mohd Za- lisham Bin Jali. Solving university examination timetabling problem using intel- ligent water drops algorithm. In Bijaya Ketan Panigrahi, Ponnuthurai Nagarat- nam Suganthan, and Swagatam Das, editors, Swarm, Evolutionary, and Memetic Computing, volume 8947 of Lecture Notes in Computer Science, pages 187–200. Springer International Publishing, 2015.
[18] Nelishia Pillay and W. Banzhaf. An informed genetic algorithm for the examination timetabling problem. Applied Soft Computing, 10(2):457–467, 2010.
[19] K.A. Dowsland and J.M. Thompson. Ant colony optimization for the examination scheduling problem. Journal of the Operational Research Society, 56:426–438, 2005.
[20] M. R. Malim, A. T. Khader, and A. Mustafa. Artificial immune algorithms for university timetabling. In E. K. Burke and H. Rudova, editors, 6th international conference on practice and theory of automated timetabling, pages 234–245, 2006.
[21] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, and Iman Yi Liao. A harmony search with multi-pitch adjusting rate for the university course timetabling. In Zong Woo Geem, editor, Recent Advances In Harmony Search Algorithm, vol- ume 270 of Studies in Computational Intelligence, pages 147–161. Springer Berlin Heidelberg, 2010.
[22] Jingpeng Li, Ruibin Bai, Yindong Shen, and Rong Qu. Search with evolutionary ruin and stochastic

rebuild: A theoretic framework and a case study on exam timetabling. European Journal of Operational Research, 242(3):798–806, 2015.

[23] Hamza Turabieh and Salwani Abdullah. A hybrid fish swarm optimisation algorithm for solving examination timetabling problems. In CarlosA.Coello Coello, editor, Learning and Intelligent Optimization, volume 6683 of Lecture Notes in Computer Science, pages 539–551. Springer Berlin Heidelberg, 2011.

[24] Nasser R. Sabar, Masri Ayob, Graham Kendall, and Rong Qu. A honey-bee mating optimization algorithm for educational timetabling problems. European Journal of Operational Research, 216(3):533 – 543, 2012.

[25] Malek Alzaqebah and Salwani Abdullah. Hybrid artificial bee colony search al- gorithm based on disruptive selection for examination timetabling problems. In Weifan Wang, Xuding Zhu, and Ding-Zhu Du, editors, Combinatorial Optimiza- tion and Applications, volume 6831 of Lecture Notes in Computer Science, pages 31–45. Springer Berlin Heidelberg, 2011.

[26] Cheng Weng Fong, Hishammuddin Asmuni, Barry McCollum, Paul McMullan, and Sigeru Omatu. A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems. Information Sciences, 283:1–21, 2014.

[27] Souad Larabi Marie-Sainte. A survey of particle swarm optimization techniques for solving university examination timetabling problem. Artificial Intelligence Review, pages 1–10, 2015.

[28] Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and Rong Qu. A survey of hyper-heuristics. Technical report, University of Nottingham, 2009.

[29] Quan-Ke Pan, M. Fatih Tasgetiren, P. N. Suganthan, and T. J. Chua. A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem. Information Sciences, 181(12):2455–2468, 2011.

[30] Gunter Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. Journal of Computational Physics, 104(1):86–92, 1993.

[31] Edmund K. Burke and Y. Bykov. The late acceptance hill-climbing heuristic. Technical report, Computing Science and Mathematics, University of Stirling, 2012.

[32] Edmund K. Burke, Graham Kendall, Mustafa Misir, and Ender O¨zcan. Monte Carlo hyper-heuristics for examination timetabling. Annals of Operations Re- search, 196(1):73–90, 2012.

[33] Nasser R. Sabar, Masri Ayob, Rong Qu, and Graham Kendall. A graph coloring constructive hyper-heuristic for examination timetabling problems. Applied Intelligence, 37(1):1–11, 2012.

[34] SyarizaAbdul-Rahman,AndrzejBargiela,EdmundK.Burke,EnderO¨zcan,Barry McCollum, and Paul McMullan. Adaptive linear combination of heuristic order- ings in constructing examination timetables. European Journal of Operational Research, 232(2):287–297, 2014.

[35] Edmund K. Burke, Rong Qu, and Amr Soghier. Adaptive selection of heuristics for improving exam timetables. Annals of Operations Research, 218(1):129–145, 2014.

[36] Salwani Abdullah and Malek Alzaqebah. A hybrid self-adaptive bees algorithm for examination timetabling problems. Applied Soft Computing, 13(8):3608–3620, 2013.