MISTA 2009

Solving Examination Timetabling Problems using Honey-bee Mating Optimization (ETP-HBMO)

Nasser R. Sabar • Masri Ayob • Graham Kendall

Abstract Examination timetabling problems deal with assigning a set of exams into a limited number of timeslots, whilst satisfying a set of constraints. In this work, we propose ETP-HBMO (Examination Timetabling Problems-Honey-bee Mating Optimization) algorithm for solving examination timetabling problems. The honey-bee mating process is considered as a typical swarm-based approach to optimization, in which the search algorithm is inspired by the process of real honey-bee mating. The mating process (i.e. generating a new solution) of the queen (current best solution) begins when the queen leaves the nest to perform a mating flight during which time the drones (trial solutions) follow the queen and mate with her. In this work, we test ETP-HBMO on Carter's un-capacitated benchmark examination timetable dataset and evaluate the performance using standard proximity costs. Results demonstrate that the performance of the Honey-bee Mating Optimization Algorithm is comparable with the results of other methodologies that have been reported in the literature over recent years. Indeed, ETP-HBMO outperformed other approaches on a few instances. This indicates that ETP-HBMO is effective in solving examination timetabling problems, and worthy of further research.

1 Introduction

Since exam timetabling problems occur frequently throughout an institution's academic calendar, it is a problem that needs to be tackled on a regular basis. Examination timetabling problems can be defined as an allocation of a number of exams to a given number of time periods (see Carter [1]). The allocation is subject to sets of hard and soft constraints. Hard constraints cannot be violated under any circumstances. For example, any exams which involve common students cannot be scheduled at the same time (conflicting exams). A feasible timetable is one in which all examinations have been assigned to feasible timeslots without violating any hard constraints. Soft constraints can be violated if absolutely necessary but every effort is made to satisfy them as far as possible. However, it is usually impossible to satisfy all the soft constraints. For example, we attempt to spread conflicting exams as far apart as possible to allow sufficient time between exams so that students have enough revision time between exams. Unfortunately, it is not always possible to provide as much time as students would ideally want. In order to minimize violations of soft constraints, a penalty cost is attached to each violation as a deterrent measure. The quality of each examination timetable is

Nasser R. Sabar
Jabatan Sains Komputer, Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi Selangor
E-mail: naserdolayme@yahoo.com

Masri Ayob
Jabatan Sains Komputer, Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia, 43600 UKM, Bangi Selangor
E-mail: masri@ftsm.ukm.my

Graham Kendall
ASAP Research Group, School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK.
E-mail: gxk@cs.nott.ac.uk

ultimately measured according to the value of penalty costs incurred. The lower the penalty, the better the quality of the exam timetable.

Over the last decade, many university exam timetabling papers have appeared in the literature. Many meta-heuristics approaches have been reported for solving examination timetabling problems. For example: graph based heuristics [2] [3], basic local search [4] [5], hybrid algorithm [6], tabu search [7] [8], simulated annealing [9], genetic algorithms [10], memetic algorithms [11] [12] great deluge algorithms [13], Ant Colony [14], particle swarm optimization [15] and Fuzzy Reasoning [16]. For interested reader, there are also a number of surveys and papers on exam timetabling [17-24].

However, as far as we are aware, there has been no work undertaken to solve examination timetabling problems utilizing a Honey-bee Mating Optimization Algorithm. This work proposes the use of such an algorithm which is tested on the Carter benchmark datasets and the results are compared with other methodologies. We demonstrate how an intelligent technique inspired by nature can be used to give good quality solutions. In order to evaluate the efficiency of the Honey-Bee Mating Optimization Algorithm, we test Honey-Bee Mating Optimization on the un-capacitated Carter's benchmark examination timetable dataset [25] (type I, [18]) and evaluate the quality using standard proximity cost [25].

2 Honey -Bees Mating Optimization Algorithm

The Honey-bee Mating Optimization Algorithm was proposed by Abbass [26] [27] in 2001. Since then it has been used for a number of different applications where it was found to outperform some better known algorithms for different applications such as: Job Shop Scheduling, Integrated Partitioning/Scheduling, Data Mining, 3-sat problem, water resources, management problems, Nonlinear constrained and Unconstrained optimization, Stochastic Dynamic Programming, Continuous Optimization, Data Mining –Clustering, and Genetic Algorithm Improvement (see [28]). However, as far as we aware, it has not been applied to examination timetabling problems. A honey-bee colony consists of the queen(s), drones, worker(s), and broods. The Honey-bee Mating Optimization Algorithm mimics the natural mating behavior of the queen bee when she leaves the hive to mate with drones in the air ([27] [29]). Each time a mating takes place, the genetic pool is enhanced to by adding sperm to the spermatheca. A heuristic (worker) is introduced to the original HBMO to improve the queen's genetic constitution. Before the mating flight begins, the queen is initialized with some energy and only ends her mating flight when the energy level drops below the threshold (which is close to zero) [29]. A queen mates with a drone probabilistically using an annealing function (Eq.1) ([26], [27]):

$$P(Q, D) = e^{\left[\frac{-\Delta(f)}{Speed(t)}\right]} \qquad (1)$$

In the above equation, P *(Q, D)* represents the probability of the drones successful mating. *Δ(f)* represents the absolute difference between the drone fitness which constitutes the trial solution and the queen's fitness or best known solution(*Δ(f)=[F(Q)-F(D)]*). *Speed (t)* refers to the queen's speed at the time of mating (time *t*). However, according to the annealing function, the queen's speed is high at the beginning of her flights; therefore the probability of mating is high, as it is when the fitness of the drone is as good as the queen's. As the mating flight continues the queen's speed and energy decays according to equations 1 and 2.

$$Speed(t+1) = \alpha \times Speed(t) \qquad (2)$$
$$energy(t+1) = \alpha \times energy(t) \qquad (3)$$

Where t € [0, 1, 2...T]

α is a factor $\in$ [0, 1], representing the *decay rate* which relates to the rate of energy and speed reduction after each transition in the mating process. Initially, the queen's speed and energy level is randomly generated (speed(0) is the initial speed). Then, a number of mating flights are undertaken. The queen moves between different states (solutions) in the allocated space according to her speed and mates with drones using Eq. 1. Once a drone passes the probabilistic decision rule and has mated successfully with the queen its sperm is added to the queen's spermatheca to generate a list of partial solutions and the queen's speed and energy are reduced according to Eq.2 and Eq.3 respectively. At the end of the mating flight the queen returns to the nest, randomly selects sperms and performs crossover and mutation to produce a brood which is enhanced by a worker (heuristic). The numbers of workers generated for the algorithm represent the number of heuristics encoded in the program. Afterwards, if the fittest brood is found to be better than the queen it replaces her, the remaining broods are destroyed and then another mating flight is initiated. The HBMO algorithm consists of three user-defined parameters namely. 1) Fertilization capacity, which corresponds to the size of the queen's spermatheca, and represents the maximum number of possible matings during a single flight. After each successful mating, the genotype of the drone is added to queen's spermatheca and the queen's spermatheca size is increased by one until the size of spermatheca is reached. 2) The number of queens to be mated. 3) The number of broods produced by the queen as a result of the mating. Figure 1 show the original Honey-bee algorithm (adopted from Abbass [27]):

```
Initialize workers
Randomly generate the queens
Apply local search to get a better queen
For a pre–defined maximum number of mating–flights.
    For each queen in the queen list
        Initialize energy, speed and position
        While the queen's spermatheca is not full and energy >0
            The queen moves between states and probabilistically chooses drones
            If a drone is selected, then
                Add its sperm to the queen's spermatheca
            End if
            Update the queen's internal energy and speed
        End while
 End for each
 Generate broods by crossover and mutation
 Use workers to improve broods
 Update workers' fitness
 While the best brood is better than the worst queen
     Replace the least–fittest queen with the best brood
     Remove the best brood from the brood list
 End while
 Kill all broods
```

Fig. 1 –Original HBMO-[27]

3 ETP-HBMO

In this work, we propose a variation of HBMO that is ETP-HBMO for solving examination timetabling problems. ETP-HBMO combines a number of different procedures. Each one represent to a different phase of the Honey-bee mating process. Firstly, we select the number of Honey-bees that would make up the population of the initial hive. In this work, the initial population was created by utilizing graph coloring heuristics (hybridizations of *Least Saturation Degree, Largest Degree first* and *Largest Enrolments First* heuristics) to construct initial solutions (see Ayob et al. [30]). The queen was chosen according to the best solution we obtained from this initial hive. The other solutions generated during this phase became the drones to be used during the mating flight (trial solutions). Secondly, we used the probabilistic rule in Eq.1 to determine which drone the queen mated with. If the mating is successful (i.e., the drone passes the probabilistic decision rule), the drone's sperm is stored in the queen's spermatheca (mating pool). Then, by crossing the drone's genotypes with the queen's, a new brood (trial solution) is formed which can be improved by conducting a local search, carried out by the workers. In this work, TEMCQ (Tabu Exponential Monte-Carlo with Counter Heuristic for Examination Timetabling (see Sabar et al. [31])) algorithm is used as a local search on broods (new trial solutions). In the Honey-bee Mating Optimization Algorithm the workers improve the broods produced from the mating flight of the queen by initiating local search procedures. Each brood chooses one worker to be subject to a local search phase, with the possibility of replacing the queen if the solution is better than the current queen. If a brood fails to replace the queen, it will become one of the drones in the queen's next mating flight. The major difference between our proposed ETP-HBMO algorithm and the original one is that the drones that have been used are then discarded and the new broods are modified to provide fresh drones for the next mating flight. This ensures that no drone's sperm can be used more than once. Table 1 shows a summary of some of the differences between our ETP-HBMO and the original one. The algorithm is shown in Figure 2 and discussed after.

TABLE 1
TABLE 1: DIFFERENCES BETWEEN OUR METHOD AND ORIGINAL ONE

Parameters	Abbass [27] (original)	Our Proposed ETP-HBMO
Drones generations methods	Drones generated randomly	Drones generated by (LS+LD+LE)
No. of queen	1	1
Local search	Greedy SAT (GSAT)	TEMCQ
mutation	Flip-flop	-

Examination Timetabling Problems-Honey-Bees Mating Optimization
(ETP- HBMO)

1-Initialization

- Define the solutions representations
- Define D, M to be population's size (number of Drones), maximum number of mating flights (number of iteration) respectively.
- Generate the initial population of the bees using (LSD+LD+LE) …. sol_i (i=1... D).
- Calculate the fitness value for each Drones f (sol_i) (i=1... D).
- Select the best (sol_i) bee and set it as the queen Q (best solution).
- Set Iter=0; W=1 number of workers; queen's spermatheca_size =0; queen_spermatheca_Max=4; population_drones_size=0; population_drones_size_Max; spermatheca_pool=0; Q=1;

2- While Iter ≤ M

- initialize energy = rand [0.5, 1], speed= rand [0.5, 1], t=0; α =0.9 // decay rate
- while energy > 0 || queen's spermatheca_size < queen_spermatheca_Max
  Select a drone sol_i from the initial populations
  Calculate Δ (f) =f (Q)-f (sol_i)
  Generate R random number= [0, 1];
  If the exp (Δ - (f) /Speed (t)) >R // *the drone passes the probabilistic condition*
  Add sperm of the drone sol_i in the spermatheca_pool (mating pool)
  Queen's spermatheca_size = queen's spermatheca_size +1;
  End if
  t=t+1; energy (t) = $\alpha \times$ energy (t-1); Speed (t) = $\alpha \times$ Speed (t-1);

End while

3-For j = 1 to queen's spermatheca_size do

- Select a Drone sperm (sol_j) from queen's spermatheca_pool;
- Generate a brood (sol_j*) by crossing the Queen's (Q) genotype with the selected Drone sperm (sol_j)
- Initiate the worker (TEMCQ). // *set local search parameters.*
- Use the selected worker (TEMCQ) to reduce the brood's (sol_j*) fitness.
- If the brood's f (sol_j**) fitter than the queen's f (*Q*) then
  Replace the queen with the brood, $Q \leftarrow sol_j$**, f (*Q*)$\leftarrow$ f (sol_j**).
- Else
  Add sol_j** to the population_broods_size.
- End if

End for j

4- If Iter = = Max then Exit; // *terminate the programs*
Else

- Iter=Iter + 1;
- Generate new drones (sol_i) by modify broods using different neighbour structures (m1, m2, m3).
- Kill all old drones and insert the new modified broods into queen's spermatheca_pool.

End else

End While
Return the Queen (Best Solution Found)

Fig.2 ETP-HBMO pseudo-code

Step 1: Initialization

A chromosome is used to represent a candidate solution sol_i to the problem where each gene in the chromosome represents a timeslot of the candidate solution. Figure 3 provides an example of the chromosome encoding:

T1	T2	T3	T4	T5	T6	T7	T8	T9
e_2	e_{11}	e_8	e_{19}	e_7	e_{14}	e_{16}	e_{10}	e_7
e_3	e_6	e_9	e_4	e_1		e_{17}	e_{12}	e_{18}
	e_7		e_{15}	e_{13}			e_{20}	e_5
	e_{12}		e_{18}				e_{21}	

Fig 3: Chromosome (complete timetable)

The algorithm starts with three user-defined parameters and one pre-defined parameter: number of workers W=1, that is, the number of heuristics encoded in the program. We use one heuristic search TEMCQ. However, the pre-defined parameter may also be used to alter the number of active heuristics if required which means that the user may choose any of them if there is more than one heuristic encoded in the algorithm. The number of queens (Q=1), and the queen's spermatheca (mating pool) size=4, which represents the maximum number of broods that can be produced by all queens. Then we define number of mating flights M, number of broods=4 and the number of drones=12. Then, we create a set of initial solutions by employing (LSD+LD+LE), sol_i, (I=1... m). After a set of initial solutions has been generated, the initial solutions are ordered according to their fitness value and the best one is set as the queen *Q*.

Step 2: Mating flight

We used Eq.1 to select the set of drones sperm (trial solutions) from the initial population to make a mating pool for possible information exchange between the best present solution (queen) and the selected sperm (trial solution). If the drones are accepted we add it to spermatheca pool.

Step 3: Breeding process

A new set of broods (new solutions) can be generated by employing pre-defined crossover operators between the queen (best solution) and drone (trial solutions) (e.g. sol_i,). In this work we use a simple crossover (haploid crossover) operation as illustrated in the example shown in Figure 4.

T1	T2	T3	T4	T5	T6	T7	T8	T9
e_2	e_{11}	e_8	e_{19}	e_7	e_{14}	e_{16}	e_{10}	e_{12}
e_3	e_6	e_9	e_4	e_1		e_{17}	e_{12}	e_{18}
	e_7	e_{15}		e_{13}			e_{20}	e_5
							e_{21}	

A: Drone

T1	T2	T3	T4	T5	T6	T7	T8	T9
e_{13}	e_8	e_5	e_{19}	e_7	e_{14}	e_{18}	e_{10}	e_7
e_3	e_6	e_9	e_4	e_1	e_{12}	e_2	e_{12}	
e_{21}		e_{11}	e_{15}	e_{20}	e_{16}		e_{17}	

B: Queen

T1	T2	T3	T4	T5	T6	T7	T8	T9
e_{13}	~~e_8~~	e_5	e_{19}	e_7	e_{14}	e_{18}	e_{10}	e_7
e_3	e_6	~~e_9~~	e_4	e_1	e_{12}	e_2	e_{12}	
e_{21}		e_{11}	~~e_{15}~~	e_{20}	~~e_{16}~~		e_{17}	
e_8							e_{16}	
e_9							~~e_{17}~~	
e_{15}								

C: brood

Fig 4: Example of haploid crossover

Two solutions are involved to produce a brood, which are a drone and a queen as shown in Figure 4(a, b). We apply haploid crossover after selecting two random genes (timeslots) from both the drone and the queen. The result of crossover might produce an infeasible solution since some of exams have been repeated. A repair mechanism has to be applied to ensure that repeated exams are removed from the brood as shown in Fig 4(c). The fitness of the trial solutions (broods) might be better than the best solution (queen). Afterwards, a local search is applied to improve the trial solutions (broods). The output from the local search is compared with the best solution. If it is better, the queen is replaced by the brood (new solution). Otherwise we keep the original queen as the best solution. The new brood will be modified using the procedure shown in Figure 5, in order to generate new drones to replace the old drones for the next mating flight.

-Define the number_of_broods to be modified

For i=1 to number_of_broods do

- Randomly move one e1 exam to another timeslots with ensuring the feasibility.
- Swap two exams e1, e2 between different timeslots with ensuring the feasibility
- Permute three exams e1, e2, and e3 in 3- cycles with ensuring the feasibility.

End of for

Fig. 5 Modified procedure pseudo-code

Step 4: Check the termination criteria
The termination criteria for ETP-HBMO algorithm is a number of mating flights.

4 Experiments and Results

In order to evaluate the performance of the ETP-HBMO algorithm, we used Carters un-capacitated exam timetabling benchmark datasets and proximity cost [25] (type I, [18]). We performed our experiments on 12 datasets (10 runs for each dataset, each run taking 2-8 hours based on the size of the dataset). The characteristics of the datasets are shown in Table 2.

The proposed algorithm was implemented in Visual C++ 6.0 on a PC AMD Athlon with a 1.92 GHz processor and 512 RAM running Windows XP 2002. In this work, we have tested a number of different parameter settings and the ones selected are those values that gave the best results with respect to solution quality and the computational time needed to achieve this solution. These parameters will be further tuned in our future work. Table 3 presents the parameter values that we have used in the current implementation

The average and best (best result out of ten runs) values are presented in Table 4. The best results obtained from literature and our works are shown in bold. Result show that ETP-HBMO is able to outperform other approaches that have been reported in the literature six instances (Car-f-92, Kfu-s-93, Sta-f-83, Tre-s-92, Uta-s-92, and Yor-f-83).

In Fig. 6, we show the box-plots of solution quality distributions. In most of cases our approach is able to produce robust solutions.

TABLE 2
UN-CAPACITATED STANDARD CARTER BENCHMARK EXAM TIMETABLING DATASET

Data sets	Number of timeslots	Number of examinations	Number of Students
Car-f-92	32	543	18419
Car-s-91	35	682	16925
Ear-f-83	24	190	1125
Hec-s-92	18	81	2823
Kfu-s-93	20	461	5349
Lse-f-91	18	381	2726
Rye-s-93	23	486	11483
Sta-f-83	13	139	611
Tre-s-92	23	261	4360
Uta-s-92	35	622	21267
Ute-s-92	10	184	2750
Yor-f-83	21	181	941

TABLE 3
ETP-HBMO PARAMETERS

No	Parameters	Value
1-	No. of Queen	1
2-	No. of Drones	12
3-	No. of Mating Flights	1000
4-	Size of Queen Spermtheca	4
5-	No. of Broods	4
6-	No. of Workers	1
7-	No. of selected gens in Crossover	8
8-	α decay rate	0.9
9-	Speed and energy	[0,1]
10-	No. of iteration for TEMCQ	5000
11-	Q TEMCQ	10
12-	Tabu Tuner TEMCQ	4

TABLE 4
RESULTS OBTAINED FROM ETP-HBMO COMPARED TO THE LITERATURE

Data sets	Carter et al. [24]	Di Gaspero and Schaerf [7]	Caramia et al. [5]	Burke and Newall [4]	Merlot et al. [6]	Kendall and Hussin [8]	Asmuni et al. [16]	White et al. [23]	Burke et al. [17]	Burke et al. [2]	Burke et al. [21]	Abdullah et al. [24]	Abdullah et al. [20]	ETP HBMO (Average)	**ETP HBMO (Best)**
Car-f-92	6.2	5.2	6.0	4.10	4.3	4.67	4.56	4.63	4.4	5.36	4.0	4.4	4.1	4.30	**3.90**
Car-s-91	7.1	6.2	6.6	4.65	5.1	5.37	5.29	5.73	4.8	4.53	**4.6**	5.2	4.8	4.86	4.79
Ear-f-83	36.4	45.7	**29.3**	37.05	35.1	40.18	37.02	45.8	35.4	37.92	32.8	34.9	36.0	36.43	34.69
Hec-s-92	10.8	12.4	**9.2**	11.54	10.6	11.86	11.78	12.9	10.8	12.25	10.0	10.3	10.8	10.84	10.66
Kfu-s-93	14.0	18.0	13.8	13.90	13.5	15.84	15.81	17.1	13.7	15.2	**13.0**	13.5	15.2	13.41	**13.00**
Lse-f-91	10.0	15.5	**9.6**	10.82	10.5	-	12.09	14.7	10.4	11.33	10.0	10.2	11.9	10.56	10.00
Rye-s-93	7.3	-	6.8	-	8.4	-	10.35	11.6	8.9	-	-	8.7	-	11.90	10.97
Sta-f-83	161.5	160.8	158.2	168.73	157.3	157.38	160.42	158	159.1	158.19	159.9	159.2	159.0	159.67	**157.04**
Tre-s-92	9.6	10.0	9.4	8.35	8.4	8.39	8.67	8.94	8.3	8.92	**7.9**	8.4	8.5	8.00	**7.87**
Uta-s-92	3.5	4.2	3.5	3.20	3.5	-	3.57	4.44	3.4	3.88	3.2	3.6	3.6	3.28	**3.10**
Ute-s-92	25.8	29.0	**24.4**	25.83	25.1	27.60	27.78	29.0	25.7	28.01	24.8	26.0	26.0	26.98	25.94
Yor-f-83	41.7	41.0	36.2	37.28	37.4	-	40.66	42.3	36.7	41.37	37.28	**36.2**	**36.2**	36.77	**36.18**

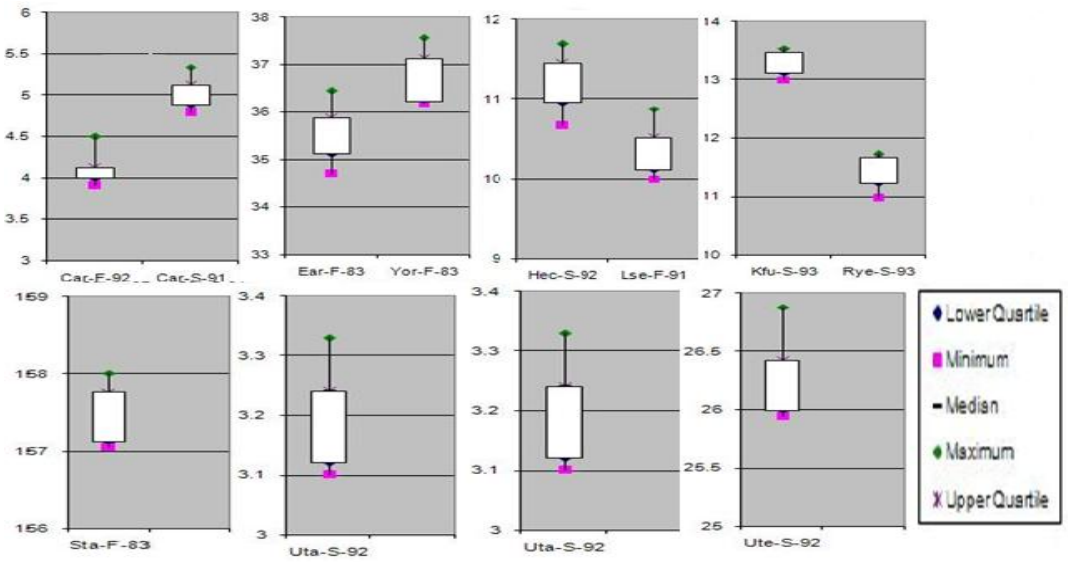

Fig 6: shows show the box-plots of solution quality distributions

4 Conclusion

In this work, a proposed Honey-bee Mating Optimization algorithm (ETP-HBMO) for solving examination timetabling problems has been presented. It has been previously been applied to other NP-hard problems but this is the first time, as far as we are aware, that it has been applied to examination timetabling. The algorithm mimics the mating flight of a queen bee. A mating flight is considered as a set of transitions, in which the queen mates with drones, probabilistically. At the start, the queen's speed is high and therefore the probability of mating is also high, which is also the case when the fitness of the drones is as good as the queen's. The

algorithms start by initializing the population and the best solution is selected as the queen. Next, a number of matings is performed. In each one, the queen's energy and speed are randomly generated. The queen then moves between different solutions in the solution space, according to her speed, and mates with the drones. If a mating is successful, the drone deposits its sperm in her spermatheca, thus giving her many genotype combinations in order to create a new brood. The worker (heuristic) is then used to improve the brood. Now the queen is replaced with the fittest of the brood. The remaining drones are then killed and the new brood is modified to be new drones for the next mating flight.

The results show that the ETP-HBMO can produce good quality solutions for benchmarks examination timetabling problems. In future work, we plan to widen our investigations to the new benchmark datasets that have recently been introduced; as well as other domains such as course timetabling.

REFERENCES

1. Carter, M.W., Laporte, G.: Recent developments in practical examination timetabling. In: Burke, E.K., Ross, P. (eds.) PATAT 1995. LNCS, vol. 1153, pp. 3–21. Springer, Heidelberg (1996).
2. Burke, E.K., McCollum, B., Meisels, A., Petrovic, S., Qu, R.: A Graph-Based Hyper-Heuristic for Educational Timetabling Problems. European Journal of Operational Research 176, 177–192 (2007).
3. de Werra, D.: An introduction to timetabling. European Journal of Operational Research 19,151-162 (1985).
4. Burke, E.K., Newall, J.: Enhancing timetable solutions with local search methods. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 195–206. Springer, Heidelberg (2003).
5. Caramia, M., Dell'Olmo, P., Italiano, G. F.: New algorithms for examination timetabling. In: Naher, S., Wagner, D. (eds.) WAE 2000. LNCS, vol. 1982, pp. 230-241. Springer, Heidelberg (2001).
6. Merlot, L.T.G., Borland, N., Hughes, B.D., Stuckey, P.J.: A hybrid algorithm for the examination timetabling problem. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 207–231. Springer, Heidelberg (2003).
7. Di Gaspero, L., Schaerf, A.: Tabu search techniques for examination timetabling. In: Burke, E.K., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, pp. 104–117. Springer, Heidelberg (2001).
8. Kendall, G. and Hussin, N.M.: Tabu Search Hyper-Heuristic Approach to the Examination Timetabling Problem at University Technology MARA. In: Burke, E.K., Trick, M. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 199-218. Springer, Heidelberg (2005).
9. Thompson, J., Dowsland, K.: A robust simulated annealing based examination timetabling system. Computers Operations Research 25, 637–648 (1998).
10. Burke, E.K., Elliman, D.G., Weare, R.F.: A hybrid genetic algorithm for highly constrained timetabling problems. In: Proceedings of the 6th International Conference on Genetic Algorithms (ICGA'95), pp. 605-610, Morgan Kaufmann, San Francisco, CA, USA (1995).
11. Burke, E.K., Newall, J.P. Weare, R.F.: A memetic algorithm for university exam timetabling. In Burke, E.k., Ross, P. (eds.) PATAT 1996. LNCS, vol. 1153, pp. 241-250. Springer, Heidelberg (1996).
12. Burke, E.K., Landa Silva, J.D.: The design of memetic algorithms for scheduling and timetabling problems. In: Hart, W. E., Krasnogor, N., Smith, J. E. (eds.) Studies in fuzziness and soft computing: Recent advances in memetic algorithms and related search technologies, vol. 166, pp. 289–312. Springer, Heidelberg (2004).
13. Burke, E.K., Bykov, Y., Newall, J.P., Petrovic, S.: A time-predefined local search approach to exam timetabling problems. IIE Transactions on Operations Engineering 36, 509–528 (2004).

14. Dowsland, K., Thompson, J.: Ant colony optimisation for the examination scheduling problem. Journal of the Operational Research Society 56, 426–439 (2005).
15. Chu, S.C., Chen, Y.T., Ho, J.H.: Timetable scheduling using particle swarm optimization. In: First international conference on innovation computing, Information and control. vol.3, pp. 324-327. IEEE Computer Society, Washington, DC, USA (2006).
16. Asmuni, H., Burke, E.K., Garibaldi, J.M., McCollum, B.: Fuzzy multiple ordering criteria for examination timetabling. In: Burke, E.K., Trick, and M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 334–353. Springer, Heidelberg (2005).
17. Burke, E.K., Kingston, J., de Werra, D.: In: Gross, J., Yellen, J. (eds.) Applications to timetabling. Handbook of Graph Theory, pp. 445–474. Chapman Hall/CRC Press (2004).
18. Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y.: A Survey of Search Methodologies and Automated System Development for Examination Timetabling. Journal of Scheduling, 12(1), 55-89 (2009).
19. Burke, E.K., Petrovic, S.: Recent research directions in automated timetabling. European Journal of Operational Research 140, 266–280 (2002).
20. Abdullah, S., Burke, E.K.: A Multi-start Large Neighbourhood Search Approach with Local Search Methods for Examination Timetabling. In Long, D., Smith, S.F., Borrajo, D., McCluskey, L. (eds.) ICAPS 2006, pp. 334-337, Cumbria, UK (2006).
21. Burke, E.K., Eckersley, A.J., McCollum, B., Petrovic S., Qu, R.: Hybrid variable neighbourhood approaches to university exam timetabling. Accepted by European Journal of Operational Research, Technical Report NOTTCS-TR-2006-2, School of Computer Science, University of Nottingham, United Kingdom (2009).
22. Ayob, M., Abdullah, S., Malik, A.M.A.: A Practical Examination Timetabling Problem at the Universiti Kebangsaan Malaysia. International Journal of Computer Science and Network Security, 7(9), 198-204 (2007).
23. White, G. M., Xie, B. S.: Examination timetables and tabu search with longer-term memory. In Burke, E.K., Erben, W. (eds.) PATAT 2000, LNCS, vol. 2079, pp. 85–103. Springer, Heidelberg (2001).
24. Abdullah, S., Ahmadi, S., Burke, E.K., Dror, M.: Investigating Ahuja–Orlins large neighbourhood search for examination timetabling. OR Spectrum 29,351–372 (2007).
25. Carter, M.W., Laporte, G., Lee, S.Y.: Examination timetabling: Algorithmic strategies and applications. Journal of Operational Research Society 47(3), 373–383 (1996).
26. Abbass, H.A.: A monogenous MBO approach to satisfiability In: Proceeding of the International Conference on Computational Intelligence for Modeling, Control and Automation, CIMCA'2001. Las Vegas, NV, USA (2001).
27. Abbass, H.A.: Marriage in honey-bee optimization (MBO): A haplometrosis polygynous swarming approach. CEC2001. pp. 207-214. Seoul, Korea (2001).
28. Baykasoùlu, Lale Özbakır and Pınar Tapkan, Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem, Source. In: Felix T. S. Chan and Manoj Kumar Tiwari (eds.) Swarm Intelligence: Focus on Ant and Particle Swarm Optimization. ISBN 978-3-902613-09-7, pp. 532, December 2007, Itech Education and Publishing, Vienna, Austria (2007).
29. Afshar, A., Haddad, Bozog, O. Marino, M. A., Adams, B. J.: Honey-bee mating optimization (HBMO) algorithm for optimal reservoir operation. Journal of the Franklin Institute 344, 452-462 (2007).
30. Ayob, M., Malik, A.M.A., Abdullah, S., Hamdan, A.R., Kendall, G., Qu, R.: Solving a Practical Examination Timetabling Problem: A Case Study. In: Gervasi, O., Gavrilova, M. (Eds.) ICCSA 2007, LNCS, vol. 4707, pp. 611–624. Part III. Springer, Heidelberg (2007).
31. Sabar, N. R., Ayob M. and Kendall G.: Tabu Exponential Monte-Carlo with Counter Heuristic for Examination Timetabling. In: proceedings of 2009 IEEE Symposium on Computational Intelligence in Scheduling (CISched2009), pp. 90-94, Nashville, Tennessee, USA (2009).