# A Honey-bee Mating Optimization Algorithm for Educational Timetabling Problems

Nasser R. Sabar[1], Masri Ayob[1], Graham Kendall[2], Rong Qu[2]

[1]Data Mining and Optimisation Research Group (DMO), Centre for Artificial Intelligent (CAIT)
Universiti Kebangsaan Malaysia, 43600 UKM, Bangi Selangor, Malaysia
naserdolayme@yahoo.com, masri@ftsm.ukm.my

[2]ASAP Research Group, School of Computer Science
The University of Nottingham, Nottingham NG8 1BB, UK.
gxk,rxq@cs.nott.ac.uk

**Abstract:** In this work, we propose a variant of the Honey-bee Mating Optimization Algorithm for solving educational timetabling problems. The honey-bee algorithm is a nature inspired algorithm which simulates the process of real honey-bees mating. The performance of the proposed algorithm is tested over two benchmark problems; exam (Carter's un-capacitated datasets) and course (Socha datasets) timetabling problems. We chose these two datasets as they have been widely studied in the literature and we would also like to evaluate our algorithm across two different, yet related, domains. Results demonstrate that the performance of the honey-bee mating optimization algorithm is comparable with the results of other approaches in the scientific literature. Indeed, the proposed approach obtains best results compared with other approaches on some instances, indicating that the honey-bee mating optimization algorithm is a promising approach in solving educational timetabling problems.

*Key words:* Timetabling; Meta-Heuristics; Honey-bee mating; Nature inspired

## 1. Introduction

Educational timetabling problems can be defined as the problem of assigning a number of events (exams/courses) to a given number of timeslots and rooms while satisfying a set of constraints (Qu et al. 2009; Lewis 2008). These constraints are usually classified into two types. Hard constraints must be satisfied in order to provide a feasible solution, whereas, soft constraints can be violated (but we try to satisfy them as far as possible). The quality of a timetable is measured based on how well the soft constraints have been satisfied.

In recent years, there has been increased research interest into swarm-based approaches and they have been found to be effective in dealing with several NP-hard problems (Yang 2008). Yang (2008) argued that the main reason for choosing swarm-based approaches is due to their ease of implementation and their flexibility (Baykasoùluet al. 2007). A number of nature inspired algorithms have been proposed including genetic algorithms, ant colony algorithms, simulated annealing and honey-bee mating algorithms. The honey-bee mating algorithm is a relatively new approach which attempts to model the natural behavior of mating in real honey bees in order to solve combinatorial optimization problems.

Although honey-bee mating algorithms have been widely applied to solve optimization and NP-hard problems (Baykasoùluet al. 2007), as far as we are aware, there has been no work undertaken to address educational timetabling problems by using a honey-bee mating algorithm. The strengths of honey-bee algorithms are their ability to simultaneously explore (probabilistically guided by the queen's transition in the space) and exploit (by employing a local search at each iteration) the problem search space. The queen (current best solution) is the dominate solution and stores different drone's genotypes in her mating pool.

She can use some parts of these genotypes to create new broods, by combining some parts of the queen genotypes with some parts of the drone's genotype. Since the queen is the fittest individual it is hoped that this will evolve superior solutions.

Motivated by the above, this work investigates variants of honey-bee algorithms for solving educational (exam and course) timetabling problems and evaluates the algorithm against other approaches that have been presented in the scientific literature. The proposed variants attempt to avoid premature convergence by maintaining population diversity. These features distinguish honey-bee algorithms from other population based algorithms that have been utilised on university timetabling problems (for example, Burke et al. 1996; Socha and Sample 2003, 2010 and Pillay and Banzhaf 2010).

The proposed method is tested against two benchmark datasets (the Carter un-capacitated dataset for exam timetabling and the Socha course timetabling dataset) and compared with the original honey-bee algorithm and other meta-heuristic methods. Results demonstrate that this nature inspired intelligent technique can be used to obtain high quality solutions for both exam and course timetabling problems.

The rest of the paper is organized as follows. Sections 2 reviews population based algorithms for education timetabling problems. The original honey-bee algorithm is presented in section 3. Our proposed approach is presented in section 4, followed by our results in section 5. Finally, concluding remarks are presented in section 6.

## 2 Problem Descriptions

In this work, the performance of the proposed algorithm is demonstrated over two benchmark problems which are exam (Carter's un-capacitated datasets) and course (Socha datasets) timetabling problems.

### 2.1 Exam timetabling problems

Exam timetabling problem can be defined as the allocation of a number of exams to a given number of time periods subject to the following set of hard and soft constraints (Carter et al. 1996 and Qu et al. 2009):

- Hard constraint: exams of common students (conflicting exams) cannot be scheduled at the same time. A feasible timetable is one in which all exams have been assigned to feasible timeslots without violating the hard constraints.
- Soft constraints: conflicting exams should be spread as far apart as possible to allow sufficient revision time between exams for students.

The quality of a timetable is given by the minimization of the soft constraint violations. The proximity cost is used to calculate the penalty cost (equation 1) (see Carter et al. 1996 and Qu et al. 2009) as follows:

- $S$ is the number of students in the problems
- $m$ is the number of exams in the problem
- $e$ is a collection exams
- $t$ represent timeslots

$$C = \sum_{k=1}^{m-1} \sum_{l=k+1}^{m} (w_i \times s_{kl}) / S, \quad i \in \{0,1,2,3,4\} \quad \ldots (1)$$

Where

- $s_{kl}$ is the number of students taking both exams $e_k$ and $e_l$, if $i=|t_k-t_l| <5$;

- $w_i=2^{|4-i|}$ is the cost of scheduling two conflicted exams $e_k$ and $e_l$ (which have common enrolled students) with $i$ timeslots apart, if if $i=|t_k-t_l| <5$, i.e. $w_0=16$, $w_1=8$, $w_2=4$, $w_3=4$ and $w_4=1$; $t_k$ and $t_l$ as the timeslot of exam $e_k$ and $e_l$, respectively.

### 2.2 Course timetabling problem

The university course timetabling problem can be defined as assigning a given number of courses to a given number of timeslots and rooms subject to a set of hard and soft constraints (Socha and Samples 2003). In this work, we have used the same model presented in Socha and Samples (2003), represented as follows:

- A set of courses $c_i$ (i = 0… $C$)
- $t_n$ represent the set of timeslots ($n$ = 1 …… 45)
- A set of $R$ rooms $r_j$ (j = 0… $R$)
- A set of $F$ room features
- A set of $M$ students

The course timetabling problem consists of assigning every course $c_i$ to a timeslot $t_n$ and room $r_j$ so that the following hard constraints are satisfied:

- No student can be assigned to more than one course at the same time.
- The room should satisfy the features required by the course.
- The number of students attending the course should be less than or equal to the capacity of the room.
- No more than one course is allowed at a timeslot in each room.

The objective is to satisfy all hard constraints and to minimize the number of students involved in the violation of soft constraints. The soft constraints are equally penalized (penalty cost = 1 for each violation per student). The soft constraints are:

- A student should not have a course scheduled in the last timeslot of the day.
- A student should not have more than two consecutive courses.
- A student should not have a single course on a day.

## 3 Related Work in Education Timetabling

Over the last two decades, meta-heuristic approaches have been successfully applied to educational timetabling problems. For example, graph based heuristics (Burke et al 2007; Sabar et al. 2009b), tabu search (Di Gaspero and Schaerf 2001), large neighbourhood search (Abdullah and Burke 2006), great deluge algorithms (Landa-Silva and Obit 2008), hybrid algorithms (Sabar et al. 2009a), and population based algorithms including memetic algorithms (Burke et al. 1996), ant colony (Socha and Sample 2003) and genetic algorithms (Pillay and Banzhaf 2010) have all been utilized. The honey-bee mating optimization (HBMO) algorithm belongs to the population-based algorithms. In this paper, we review the population based algorithms that have been applied to university timetabling problems. Interested readers are referred to recent surveys in this area (Qu et al. 2009; Lewis 2008; McCollum et al. 2010; and Burke and Petrovic 2002) for more comprehensive coverage of other methodologies.

### 3.1 Population Based Algorithms for Exam Timetabling

Population based algorithms, such as genetic algorithms and ant colony algorithms, have been utilized to solve exam timetabling problems. Cote et al. (2005) proposed a bi-objective evolutionary algorithm to minimize the timetable length and to space out conflicting exams as much as possible. The recombination

operators were replaced by two local searches (tabu search and variable neighbourhood descent) to deal with hard and soft constraint violations. The methods obtained competitive results on a number of benchmark problems. However, replacing the crossover and mutation operators by two local searches led to an increased number of parameters that needed to be tuned, which is one of the main disadvantages for many meta-heuristic approaches.

Eley (2007) applied two ant algorithms to simultaneously construct and improve exam timetables. The first algorithm, MMAS-ET, is based on the Max-Min Ant System that was used by Socha and Samples (2003) on course timetabling problems. The second algorithm ANTCOL-ET is a modified version of ANTCOL (originally used by Costa and Hertz (1997) to solve graph colouring problems). Both ant algorithms were hybridized with a hill climber and tested on the Carter benchmark datasets. Results showed that the simple ant system ANTCOL-ET outperformed the more complex MMAS-ET. Indeed, the performance of ant systems can be considerably improved by adjusting the search parameters such as the evaporation rate, the pheromone deposit interval and the number of cycles. However, in the construction stage (exploration), the ants are trying to generate a feasible timetable from scratch by using previous knowledge (pheromone). Therefore, the algorithm may struggle to obtain a feasible timetable in some cases if parameter values are not set appropriately, especially for highly constrained problems.

Ersoy et al. (2007) proposed a combination of hill-climbing and memetic algorithms in a hyper-heuristic framework. The authors compared their approach with a self-adaptive memetic algorithm based hyper-heuristic with different heuristic selection and acceptance criteria. The experimental results on the Carter benchmark datasets showed that a memetic algorithm based hyper-heuristic, which used a single hill climber at a time, performed the best among variants of other hill climbing hyper-heuristics proposed by the same authors.

Burke et al. (2010) presented a variant of variable neighbourhood search for solving exam timetabling problems. An indirect genetic algorithm was employed to intelligently select a subset of neighborhoods. Good results were obtained on some of the Carter benchmark problem instances.

Recently, Pillay and Banzaf (2010) proposed a two-stage informed genetic algorithm for exam timetabling problems. The first phase focuses on generating feasible timetables that satisfy all hard constraints, whilst the second phase tries to minimize soft constraint violations. In both phases, a genetic algorithm was used to generate and improve the timetable. Comparable results have been obtained on the Carter benchmark datasets over other approaches. However, they did not mention the rationale behind using a GA in the construction and improvement phases, which increases the computational time and the number of parameters that need to be tuned. Furthermore, all obtained results are worse than the best known.

### 3.2 Population Based Algorithms for Course Timetabling

Socha and Samples (2003) used ant colony optimization to solve university course timetabling problems. They compared two algorithms, ant colony system and a MAX-MIN algorithm. At each step, each ant constructs a complete solution that meets all the hard constraints by using heuristics and pheromone information. Then, a hill climbing local search is used to improve the solutions. Both approaches were tested on the same datasets that were originally introduced by the same authors. MAX-MIN system usually achieves better results.

Abdullah et al. (2007) presented a hybrid evolutionary approach to university course timetabling problems. Starting with feasible solutions, a memetic approach with a mutation operator and a randomized iterative improvement technique is used to improve the solutions. Roulette wheel selection is utilized to select the individuals, which are improved by a local search. The proposed method, tested on the Socha dataset, produced good quality results for all tested datasets. Ignoring the crossover operator (the main operator in a GA) led to an inefficient exploration of the search space.

Abdullah and Turabieh (2008) proposed a standard genetic algorithm with hill climbing for university course timetabling problems. The authors focused on mechanisms to repair infeasible solutions after applying crossover and mutation. Comparable results were obtained on the Socha datasets, but they were worse than the best known results and also those reported in Abdullah et al. (2007).

Landa-Silva and Obit (2009) proposed an evolutionary non-linear great deluge algorithm for university course timetabling. Individuals are selected by tournament selection and improved using a mutation operator. The improved individual, if better than the worst individual, replaces the worse individual in the population. Experimental results show that the hybridization between the non-linear great deluge and evolutionary operators produces good quality results on the Socha datasets. The proposed method is similar to Abdullah et al. (2007) in that using a population of solutions and applying mutation operators only, omitting the main GA operator crossover. This may restrict its ability to explore broader regions of the search space.

Turabieh et al. (2009) proposed an electromagnetism-like mechanism with force decay rate great deluge algorithm for university course timetabling. It is based on an attraction-repulsion movement for solutions in the search space. The proposed method begins with a population of randomly generated, feasible timetables and an attraction-repulsion mechanism is used to calculate the estimated quality for the great deluge algorithm. Then, a great deluge algorithm is employed to enhance the solution. This method is able to obtain very good results for the Socha datasets. However, since the estimated quality is changeable, based on the current attraction-repulsion value, it might blindly explore the search space. Also, the authors did not mention what will happen when the great deluge algorithm is unable to improve the solutions any longer.

Motivated by the above, this work proposes another population based approach, which is a variant of the HBMO algorithm. The proposed approach is designed to simultaneously explore (guided by the queen's transition in the space) and exploit (employing local search at each iteration). The differences between HBMO and the previous population based algorithms that have been used to solve university timetabling problems are:

1. Our approach is an improvement based method using a direct representation and employing a crossover operator, whereas most of the previous population based algorithms omit the crossover operator (we suspect, due to the complexity in maintaining feasibility) except (Abdullah and Turabieh 2008 and Pillay and Banzaf 2010).
2. HBMO explores the search space probabilistically, guided by the queen's transition, based on an annealing function.
3. Instead of selecting two parents, using the same selection procedure (as in a genetic algorithm), the first parent (the queen) is always the fittest individual and the second (the drone) is selected based on its fitness using Equations (1) and (2) (see section 3).
4. Our approach employs an exploitation stage, at each iteration, by applying a local search algorithm. Most previous population based algorithms did not apply an exploitation stage except for (Abdullah et al. 2007; Landa-Silva and Obit 2009; Abdullah and Turabieh 2008 and Turabieh et al. 2009).

Table 1 summaries the differences and similarities between our proposed HBMO algorithm and previous population based algorithms that have been applied to university timetabling problems.

Table 1
Differences and similarities between the HBMO algorithm and previous population based algorithms. "-" means the method did not use the corresponding operator.

| Approaches | Application Type | Solution Representation | Same Representation as | Initialization Method | Initial Solution | Crossover Type | Mutation Type | Problem Type | Exploitation during |
|---|---|---|---|---|---|---|---|---|---|

| | | | HBMO | | | | | | search |
|---|---|---|---|---|---|---|---|---|---|
| HBMO | Improvement | direct | | Graph coloring | feasible | Haploid | Shaking procedure | Exam and course | Yes (at every iteration) |
| GA (Cote et al. 2005) | Constructive + improvement | direct | NO | Random | infeasible | Local search | Local search | exam | NO |
| ACO (Eley 2007) | Constructive | direct | NO | Graph coloring | - | - | - | exam | NO |
| GA (Erosy et al. 2007) | improvement | direct | NO | Graph coloring | infeasible | - | Move operator | exam | NO |
| GA (Burke et al. 2010) | improvement | indirect | NO | Graph coloring | feasible | One point | | exam | NO |
| GA (Pillay and Banzaf 2010) | Constructive + improvement | direct | NO | GA | feasible | One point | Move operator | exam | NO |
| ACO (Socha and Samples 2003) | Constructive | direct | NO | - | feasible | - | - | Course | NO |
| GA (Abdullah et al. 2007) | improvement | direct | NO | Random+ Graph coloring | feasible | - | Move operator | Course | Yes |
| GA (Abdullah and Turabieh 2008) | improvement | direct | Yes | random | feasible | one point | Move operator | Course | Yes |
| GA + GD (Landa-Silva and Obit 2009) | improvement | direct | NO | Graph coloring | feasible | - | Great deluge | Course | Yes |
| EM+GD (Turabieh et al. 2009) | improvement | direct | Yes | Graph coloring | feasible | - | Great deluge | Course | Yes |

## 4 The Honey-Bee Mating Optimization Algorithm

The honey-bee mating optimization (HBMO) algorithm was proposed by Abbass (2001a, 2001b). It has been successfully applied to solve job shop scheduling, integrated partitioning/scheduling, data mining**,** 3-sat, nonlinear constrained and unconstrained optimization, stochastic dynamic programming and continuous optimization problems (see Baykasoùluet al. 2007). However, as far as we are aware, it has not been investigated in the context of educational timetabling.

A honey-bee colony consists of queen(s) (best solution), drones (incumbent solutions), worker(s) (heuristic), and broods (trial solutions). The HBMO algorithm simulates the natural mating behaviour of the queen bee when she leaves the hive to mate with drones (Abbass 2001a, 2001b). After each successful mating, the drone's sperm is added to the queen's spermatheca. Before the mating flight begins, the queen is initialized with some energy and only ends her mating flight when her energy level drops below a threshold (which is close to zero) (Afshar et al., 2007). Table 2 illustrates the analogy between the natural honey bee colony and the artificial honey bee algorithm. Figure 1 presents the pseudo-code of the original honey-bee algorithm (adopted from Abbass 2001a).

Table 2
Analogy between the natural honey bee colony and the artificial honey bee algorithm

| Natural honey bee | Artificial honey bee |
|---|---|
| Queen | Best solution |
| Drones | Incumbent solutions |
| Broods | New trial solutions |

| Worker | Heuristic search |
|---|---|
| Mating, Breeding | Crossover |

The queen mates with a drone probabilistically, using Equation (2) (Abbass 2001a, 2001b).

$$p(Queen, Drone_i) = e^{\left[\frac{-\Delta(f_i)}{energy(t)}\right]} \quad (2)$$

where $P(Queen, Drone_i)$ represents the probability of accepting the $i^{th}$ drone for mating. $\Delta(f)$ represents the absolute fitness difference between the drone and the queen, i.e. $\Delta(f_i)=|f(Queen)-f(Drone_i)|$. $energy(t)$ refers to the queen's energy at time $t$ of mating. The term fitness function in timetabling problems represents the objective function. The objective function represents the solution quality which calculates the soft constraints violations (see results section). According to Afshar et al. (2007), the queen's energy is high at the beginning of her flights (indicating that the possibility of mating is high). The possibility of mating is also high when the fitness of the drone is as good as the queens. As the mating flight continues, the queen's energy and speed decays according to Equations (3) and (4) (Abbass 2001a, 2001b).

$$energy(t+1) = \alpha \times \text{energy(t)} \qquad \text{where } t \in [0,1,2...t] \text{ and decay rate } \alpha \text{ within } [0,1] \quad (3)$$

$$speed(t+1) = \text{energy(t)} - \beta \qquad \text{where } t \in [0,1,2...t] \text{ and decay rate } \beta \text{ within } [0,1] \quad (4)$$

where $\alpha$ represents the decay rate, and relates to the rate of energy reduction after each transition in the mating process. Initially, the queen's energy level is randomly generated. Then, a number of mating flights are undertaken. The queen moves between different states (i.e. solutions) in the allocated space, according to her energy, and mates with drones using Equation (2). Once a drone has mated with the queen, its sperm is added to the queen's spermatheca. After each encounter, the queen updates her energy and speed using Equations (3) and (4). The queen ends her mating flight when her energy level drops below a threshold (which is close to zero) or the queen's maximum spermatheca size is reached.

```
Initialize worker
Randomly generate an initial population and set the best individual as the queen
For a pre-defined maximum number of mating flights
    Initialize the queen's energy and speed randomly
    While queen's energy > 0
        The queen moves between states (solutions) and chooses drones probabilistically using Equation (2)
        If a drone is selected, then
            Add its sperm to the queen's spermatheca
        End if
        Update the queen's internal energy and speed using Equations (3) and (4)
    End while
    Generate broods by applying crossover and mutation
    Use the worker to improve the broods
    If the best brood is fitter than the queen then
        Replace the queen with the best brood
    End if
  Kill all broods
End for
```

Figure 1 The Original HBMO Algorithm (Abbass 2001a)

At the end of the mating flight, the queen returns to the nest. Then, the queen starts breeding by randomly selecting a drone's sperm from her spermatheca and performs crossover to produce a brood (one brood per crossover) which is then fed by a worker to enhance the broods (all new generated broods are en-

hanced by a worker). The number of workers used for the algorithm represents the number of heuristics. If the fittest brood is superior to the queen, it replaces her. All other broods, and the former queen, are destroyed and then another mating flight is initiated, with a new queen and the same pool of drones.

The honey-bee mating algorithm shares two operators (crossover and mutation) with genetic algorithms but has two main differences. Firstly, HBMO always uses the queen (which is the dominant solution) and stores different drone's genotypes in her mating pool in order to create new broods by combining some parts of the queen genotypes with the same parts of the drone's genotype. Since the queen is the fittest individual, we hope that this will evolve better solutions. This kind of combination leads the search towards better regions of the search space. In comparison, in genetic algorithms two parents are selected to produce new children. This may not be effective in exploring the search space since the selection mechanism employs probabilistic and random factors (e.g. roulette wheel selection). Secondly, HBMO applies a local search at every iteration, which can be considered as an exploitation stage. By comparison there is no local search (exploitation) in standard genetic algorithms, although a memetic algorithm can be used to incorporate this feature.

## 5 The Honey-Bee Mating Optimization Algorithm for Educational Timetabling Problems

In this work, we propose a variation of HBMO, which we refer to as HBMO-ETP, for solving educational timetabling problems. The proposed HBMO-ETP is an improvement based method which starts with populations of feasible solutions, followed by the HBMO mating process.

Firstly, we select a number of honey-bees to create the population of the initial hive. Previous work (Qu et al. 2009; Lewis 2008; Sabar et al. 2009a; Sabar et al. 2009b and Ayob et al. 2007) showed that in many cases, random generation methods may not necessarily guarantee a good quality or even feasible solution in some cases. Therefore, in this work, we employ hybrid graph coloring heuristics (Ayob et al., 2007), to generate an initial population of feasible solutions. The three graph coloring heuristics we utilize are:

- Least Saturation Degree First (SD): events are ordered dynamically, in an ascending order, by the number of remaining available feasible timeslots.
- Largest Degree First (LD): events are ordered, in a decreasing order, by the number of conflicts they have with all other events.
- Largest Enrolment First (LE): events ordered by the number of students enrolled, in a decreasing order.

The generation method starts with an empty timetable and applies the hybridized heuristics to schedule the unscheduled events. The hybridized heuristic (SD+LD+LE) sorts the unscheduled events in a non-decreasing order of the number of available timeslots (SD). Those with equal SD evaluations are then arranged in a non-increasing order of the number of conflicts they have with other events (LD) and those with equal LD evaluations are then arranged in a non-increasing order of the number of student enrolments (LE). Then, the first event in the unscheduled list is selected to be assigned to a timeslot that satisfies all the hard constraints. We assign events to a random timeslot when the event has no conflict with those that have already been scheduled, ensuring that all hard constraints are satisfied. The process of selection and assignment of events is repeated until all events have been scheduled, or some events cannot be assigned to any available timeslot. If this occurs, we stop the process and start again. Although there is no guarantee that a feasible solution can be generated, for all the instances used in this work, we were always able to obtain a feasible solution. The best solution in this initial population becomes the queen. The other solutions generated during this phase become the drones.

Next, we use equation (2) to determine which drone the queen will mate with. Based on our preliminary experiments we eliminated the speed parameter (equation (4) in the original HBMO, (see section 4) since it did not affect the selection of a drone to mate with the queen. Therefore, we only use the energy

equation (3). Note that the elimination of the speed parameter will decrease the number of parameters that need to be tuned (i.e. decay rate in Equation (4)). If the mating is successful (according to the probabilistic decision rule), the drone's sperm is added into the queen's spermatheca.

Next, the queen starts breeding and some broods are formed. A worker is then applied (i.e. an improvement) to the resultant broods. We utilize a simple descent algorithm as the worker to improve the trial solutions. As in the original honey-bee mating optimization algorithm, the workers improve the brood produced from the breeding queen with the possibility of replacing the queen if the improved brood is better than the current queen.

The original HBMO (see section 4) suffers from premature convergence. This is due to the initial population never being updated or modified during the mating and breeding process. In order to avoid premature convergence, the population of solutions is updated at every mating process. The major difference between our proposed HBMO-ETP algorithm and the original is that all the drones that have been used are then discarded and the new broods are modified to provide fresh drones for the next mating flight. In the original HBMO, all broods are killed and the new mating flight begins using the previous population. This ensures that no drone's sperm can be used more than once which, we hope, maintains diversity and stops premature convergence. Table 3 shows a summary of differences and similarities between our HBMO-ETP and the original algorithm. The pseudo-code for the HBMO-ETP algorithm is shown in Figure 2.

Table 3
Differences and similarities between our HBMO-ETP and the original HBMO

| Parameters | Abbass (2001a) Original HBMO | Our Proposed HBMO-ETP |
|---|---|---|
| Drones generation methods | Generated randomly | Generated by (LS+LD+LE) |
| No. of queens | 1 | 1 |
| Local search | Greedy SAT (GSAT) | Simple Descent |
| Crossover | haploid | haploid |
| Mutation type | Flip | Shaking procedure |
| Resultant broods | All broods are killed | All broods will be used in the next mating flight |
| Fitness function | Fitness function | Objective function (timetable quality or penalty cost) |

```
1.  Set the maximum number of mating flights M, the number of queens =1, Iter = 0
2.  Set the number of workers W; queen spermatheca q_s = Ø
3.  Set the maximum queen spermatheca size q_s_mx
4.  Set drones population d_pop = Ø; drones maximum population size d_mx_pop
5.  Set broods population b_pop = Ø;
6.  For i = 1 to  d_mx_pop
7.      Generate the drone sol_i and add it to d_pop
8.  End for
9.  Calculate the fitness value for each drone f(sol_i)
10. Select the best drone (sol_i) and set it as the queen Q (best solution)
11. While Iter ≤ M
12.     Iter = Iter + 1
13.     Initialize energy = rand [0.5, 1], t = 0;  α = 0.9
14.     While energy > 0 or |q_s| < q_s_mx
15.         Randomly select a drone sol_i from d_pop
16.         Calculate Δ(f) = |f(Q)-f(sol_i)|
17.         Generate a random number R in range (0, 1)
18.         If exp (-Δ(f) /energy (t)) < R
19.             Add sperm of the drone sol_i to q_s
20.             |q_s| = |q_s| +1
21.         End if
22.         t = t + 1; energy (t) =  α ×  energy (t-1)
23.     End while
24.     For j = 1 to |q_s| do
25.         Select a drone sperm (sol_j) from q_s
26.         Generate a brood (sol_j*) by crossing the queen's (Q) genotype with the selected drone sperm (sol_j) using
              haploid crossover
27.         Apply the selected worker (Simple Descent) to improve the brood's (sol_j*) fitness
28.         If the best brood's f(sol_j*) is fitter than the queen's f(Q) then
29.             Replace the queen with the brood, Q= sol_j* and f(Q) =f (sol_j*)
30.         Else add sol_j* to b_pop
31.         End if
32.     End for j
33.     Shake all broods (b_pop) by using Kemp-chain neighbor structures
34.     Kill all old drones and insert the new shaked broods (b_pop) into d_pop
35. End While
36. Return the queen (Best Solution Found)
```

Figure 2 The pseudo code of HBMO- ETP

In lines 1 to 5 (see figure 2), we initialize three user-defined parameters. These are: (i) the number of queens, (ii) the queen's spermatheca size, which represent the maximum number of matings each queen performs in a single mating flight, thus also the number of broods that will be born after each single mating flight and (iii) the number of workers. We only use one worker (heuristic search) i.e. simple descent algorithm. However, any other local search is also applicable. In lines 6 to 8, we create the drone population by employing hybridizations of graph colouring (*Least Saturation Degree, Largest Degree first* and *Largest Enrolments First*) heuristics (see Ayob et al. 2007). The best solution is set as the queen *Q* in lines 9 and 10.

Lines 11 to 35 represent a mating flight. In line 13, we initialize the queen with energy. Then, we select the set of drones from the drone population based on Equation (2) to build a mating pool for possible information exchange between the queen and the selected drone in line 15. If the drones are accepted we add them to the queen's spermatheca (lines 16 to 21). Based on Equation (2), the fitter drones have more chance of being selected. This procedure is repeated until the queen's maximum spermatheca size is reached or the queen's energy drops to zero. Then, the breeding process starts (lines 24 to 33). A new set

of broods can be generated by employing pre-defined crossover operators between the queen and drones (see line 26).

In our HBMO-ETP algorithm, a chromosome is used to represent a candidate solution $sol_i$ to the problem where each gene represents a timeslot of the candidate solution. Table 4 provides an example of the chromosome encoding, where columns represent timeslots e.g. $T_1$, $T_2$...$T_9$ and rows represent rooms. The entries of Table 4 are the events themselves, e.g. $e_2$, $e_3$, $e_4$ … $e_n$.

Table 4 chromosome (complete timetable)

| | **Timeslots** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Rooms** | **$T_1$** | **$T_2$** | **$T_3$** | **$T_4$** | **$T_5$** | **$T_6$** | **$T_7$** | **$T_8$** | **$T_9$** |
| | $e_2$ | $e_{11}$ | $e_8$ | $e_{19}$ | $e_7$ | $e_{14}$ | $e_{16}$ | $e_{10}$ | $e_7$ |
| | $e_3$ | $e_6$ | $e_9$ | $e_4$ | $e_1$ | | $e_{17}$ | $e_{12}$ | $e_{18}$ |
| | | $e_7$ | | $e_{15}$ | $e_{13}$ | | | $e_{20}$ | $e_5$ |
| | | $e_{12}$ | | $e_{18}$ | | | | $e_{21}$ | |

In Figure 3, we illustrate a simple crossover (haploid crossover Abbass 2001a, 2001b) by using an example. Assume two solutions A and B, which are a drone and a queen, are employed to produce a brood. Two random genes (i.e. timeslots) from both the drone and the queen are selected (shown as the shaded genes $T_1$ and $T_8$ in chromosome (A) and genes $T_3$ and $T_7$ in chromosome (B)). Then we move all events (exams or courses) from gene $T_3$ in chromosome (B) to gene $T_1$ in chromosome (A), and from gene $T_7$ in chromosome (B) to gene $T_8$ in chromosome (A). If an event conflicts with those in the new gene, or the same event is already in the new gene (e.g. $e_{15}$ in $T_3$ chromosome (B)) the event will not be moved. A repair mechanism has to be applied to remove duplicate events, as shown in Figure 3 (C), and to make sure that the broods which are produced are feasible. All the old events, that caused duplication, are removed. If the events conflict with those in the new gene, or the same events are already in the new gene (e.g. $e_{15}$ in $T_3$ chromosome (B)) the events will not be moved. Otherwise, we move events to the new gene and delete it from the old gene as shown in Figure 3(C).

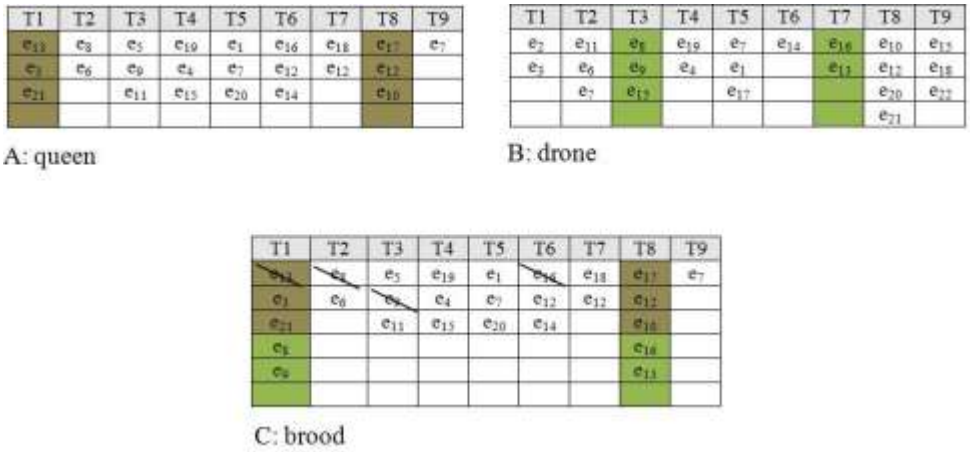


Figure 3 Example of Haploid Crossover

After crossover, a local search is applied to improve the broods. In this work, we employ a simple descent algorithm as our local search procedure. The simple descent algorithm starts with a feasible initial solution (i.e. brood) and iteratively improves it by examining its neighborhood. A neighborhood of a given solution is obtained by moving one event from its current timeslot to another timeslot, which is selected randomly. The solution is accepted, if the move does not violate any hard constraints and the quality of the neighborhood solution is better than the incumbent solution. Otherwise, the solution is rejected and a

new event is selected to generate a neighborhood solution. This process is repeated until the stopping condition is satisfied. In this work, the stopping condition is set to 5,000 iterations which was determined by our preliminary testing.

If the improved brood is better than the queen, the queen is replaced by the brood. Otherwise we keep the original queen as the best solution. In order to ensure that no drone's sperm can be used more than once, the new broods will be modified using a shaking procedure (mutation operator). In this work, we employed a kempe chain neighbourhood (see Thomson and Dowsland 1996) as a shaking procedure to modify the generated broods. The kempe chain neighbourhood is applied to each brood to swap a subset of events between two timeslots. The main feature of the kempe chain neighbourhood is the capability of moving a *chain* of events within the timetable while ensuring the feasibility. The modified broods will replace the old ones for the next mating flight.

## 6 Experimental Results and Discussion

We tested our HBMO-ETP algorithm on the un-capacitated exam timetabling problems introduced by Carter et al. (1996) and the course timetabling problems in Socha and Sample (2003) (see section 2). These two benchmarks have been widely used in the literature to evaluate the performance of different approaches. 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.

### 6.1 The HBMO-ETP Parameters Setting

The parameter settings of HBMO-ETP for both the exam and course timetabling problems are listed in Table 5. The parameters of the proposed algorithm were selected based on our preliminary experiments. They provide a good trade-off between solution quality and the computational time needed to reach good quality solutions. Our preliminary tests show that increasing the size of queen spermtheca has no impact on the algorithm performance, but the computational time is increased. We found that the most sensitive parameter is the number of selected genes for the crossover operator. Increasing the number of selected genes will increase the exploitation stage to generate a good quality solution but, at the same time require more computational time to maintain solution feasibility. Please note that we used the same parameter setting across instances within both exam and course timetabling problems. We believe that this is an important element of this paper.

Table 5 HBMO-ETP parameters

| No. | Parameters | Tested Range | Suggested Value |
|---|---|---|---|
| 1- | No. of Drones | 10-60 | 40 |
| 2- | No. of Mating Flights | 5000-15000 | 10000 |
| 3- | Size of Queen Spermtheca | 5-30 | 10 |
| 4- | No. of Broods | 5-30 | 10 |
| 5- | No. of selected genes in Crossover | 2 - (number of gens-1) | 8 |
| 6- | Simple Descent Iteration | 1000-8000 | 5000 |

### 5.2 The HBMO-ETP for the Carter Uncapacitated Exam Benchmark Dataset

Table 6 presents the characteristics of the uncapacitated exam timetabling benchmark problem instances from Carter et al. (1996) (Toronto *b* type I in Qu et al. 2009).

Our first experiment compares the performance of HBMO-ETP with the original algorithm (using equation (4)). Both HBMO-ETP and the original HMBO use the same parameter settings (Table 5). Twenty

independent runs (each taking 2-8 hours depending on the size of the problem instance) were carried out for each of the 13 instances using different random seeds. We note that this run time is acceptable in university timetabling problems because the timetables are usually produced months before the actual schedule is required (Burke et al. 2010). The results of average and best penalty cost (out of 20 runs) of HBMO-ETP and the original HBMO are presented in Table 7.

Table 6 Carter's un-capacitated benchmark exam timetabling dataset

| Data sets | Number of timeslots | Number of exams | Number of Students |
|---|---|---|---|
| Car-f-92-I | 32 | 543 | 18419 |
| Car-s-91-I | 35 | 682 | 16925 |
| Ear-f-83-I | 24 | 190 | 1125 |
| Hec-s-92-I | 18 | 81 | 2823 |
| Kfu-s-93 | 20 | 461 | 5349 |
| Lse-f-91 | 18 | 381 | 2726 |
| Pur-s-93-I | 43 | 2419 | 30032 |
| Rye-s-93 | 23 | 486 | 11483 |
| Sta-f-83-I | 13 | 139 | 611 |
| Tre-s-92 | 23 | 261 | 4360 |
| Uta-s-92-I | 35 | 622 | 21267 |
| Ute-s-92 | 10 | 184 | 2750 |
| Yor-f-83-I | 21 | 181 | 941 |

Table 7
HBMO-ETP compared against HBMO. Best penalty costs are in bold.

| Datasets | HBMO-ETP | | Original HBMO | |
|---|---|---|---|---|
| | Ave | Best | Ave | Best |
| Car-f-92-I | 4.30 | **3.90** | 4.67 | 4.19 |
| Car-s-91-I | 4.86 | **4.79** | 5.43 | 5.12 |
| Ear-f-83-I | 36.43 | **34.69** | 38.21 | 3.51 |
| Hec-s-92-I | 10.84 | **10.66** | 11.27 | 10.90 |
| Kfu-s-93 | 13.41 | **13.00** | 14.25 | 13.43 |
| Lse-f-91 | 10.56 | **10.00** | 10.83 | 10.54 |
| Pur-s-93-I | 6.3 | **4.76** | 9.48 | 8.29 |
| Rye-s-93 | 11.90 | **10.97** | 13.59 | 12.17 |
| Sta-f-83-I | 159.67 | **157.04** | 158.28 | 157.82 |
| Tre-s-92 | 8.00 | **7.87** | 8.3 | 8.00 |
| Uta-s-92-I | 3.28 | **3.10** | 3.41 | 3.29 |
| Ute-s-92 | 26.98 | **25.94** | 27.23 | 26.85 |
| Yor-f-83-I | 36.77 | **36.15** | 38.22 | 37.14 |

With reference to Table 7, the penalty cost produced by HBMO-ETP is better than the original algorithm. We believe this is due to the fact that HBMO-ETP enhances the diversity of the population by discarding the mated drones and inserting broods into the drone population.

Our second experiment compares HBMO-ETP with other meta-heuristic methodologies, both population based and local search methods. The five population based methods that we compare against in Tables 8 and 9 are:

- H1: Cote et al. (2005): Bi-objective evolutionary algorithm with local search.
- H2: Burke et al. (2010): Variable neighbourhood genetic algorithm.
- H3: Eley (2007): Ant algorithm with hill climbing.
- H4: Ersoy et al. (2007): Memetic algorithm based hyper-heuristics.
- H5: Pillay and Banzhaf: (2010): Informed genetic algorithm.

As shown in Tables 8 and 9, HBMO-ETP obtains competitive results when compared against all the population based methodologies. Moreover, for 6 instances, HBMO-ETP outperforms, or obtains the same best results, as other methods. HBMO-ETP obtained the second best results on Car-s-91 and Rye-s-93 across all population methods, whilst on Ear-f-83, Hec-s-92 and Ute-s-92 it is the third best overall population method. We can conclude that HBMO-ETP is generally able to produce high quality results when compared against other population based methods.

Table 8
HBMO-ETP compared against other population based methods in the literature. Best costs are in bold.

| Datasets | HBMO-ETP | | H1 | | H2 |
|---|---|---|---|---|---|
| | Ave | Best | Ave | Best | Best |
| Car-f-92-I | 4.30 | **3.90** | 4.4 | 4.2 | **3.9** |
| Car-s-91-I | 4.86 | 4.79 | 5.5 | 5.2 | **4.6** |
| Ear-f-83-I | 36.43 | 34.69 | 35.6 | 34.2 | **32.8** |
| Hec-s-92-I | 10.84 | 10.66 | 16.5 | 10.2 | **10.0** |
| Kfu-s-93 | 13.41 | **13.00** | 14.4 | 14.2 | **13.0** |
| Lse-f-91 | 10.56 | **10.00** | 11.5 | 11.2 | **10.0** |
| Pur-s-93-I | 6.3 | **4.76** | - | - | - |
| Rye-s-93 | 11.90 | 10.97 | 9.1 | **8.8** | - |
| Sta-f-83-I | 159.67 | 157.04 | 157.6 | 157.2 | **156.9** |
| Tre-s-92 | 8.00 | **7.87** | 8.8 | 8.2 | **7.9** |
| Uta-s-92-I | 3.28 | **3.10** | 3.6 | 3.2 | 3.2 |
| Ute-s-92 | 26.98 | 25.94 | 25.5 | 25.2 | **24.8** |
| Yor-f-83-I | 36.77 | 36.15 | 37.5 | 36.2 | 34.9 |

Table 9
HBMO-ETP compared against other population based methods in the literature. Best penalty costs are in bold.

| Datasets | HBMO-ETP | | H3 | | H4 | H5 |
|---|---|---|---|---|---|---|
| | Ave | Best | Ave | Best | Best | Best |
| Car-f-92-I | 4.30 | **3.90** | 4.4 | 4.3 | - | 4.2 |
| Car-s-91-I | 4.86 | 4.79 | 5.3 | 5.2 | - | 4.9 |
| Ear-f-83-I | 36.43 | 34.69 | 38.5 | 36.8 | - | 35.9 |
| Hec-s-92-I | 10.84 | 10.66 | 11.4 | 11.1 | 11.7 | 11.5 |
| Kfu-s-93 | 13.41 | **13.00** | 14.9 | 14.5 | 15.8 | 14.4 |
| Lse-f-91 | 10.56 | **10.00** | 11.7 | 11.3 | 13.3 | 10.9 |
| Pur-s-93-I | 6.3 | 4.76 | 4.6 | 4.6 | - | **4.7** |
| Rye-s-93 | 11.90 | 10.97 | 10.0 | 9.8 | - | 9.3 |
| Sta-f-83-I | 159.67 | 157.04 | 157.5 | 157.3 | 157.9 | 157.8 |
| Tre-s-92 | 8.00 | **7.87** | 8.7 | 8.6 | - | 8.4 |
| Uta-s-92-I | 3.28 | **3.10** | 3.5 | 3.5 | - | 3.4 |
| Ute-s-92 | 26.98 | 25.94 | 27.5 | 26.4 | 26.7 | 27.2 |
| Yor-f-83-I | 36.77 | 36.15 | 40.7 | 39.4 | 40.7 | 39.3 |

Tables 10 and 11 presents the penalty cost of HBMO-ETP compared to other (non-population based) methods in the literature. These approaches we compare against are:

- P1: Carter et al. (1996): Largest cliques as the initialization for graph heuristics with backtracking.
- P2: Di Gaspero and Schaerf (2001): Tabu search algorithm.
- P3: Caramia et al. (2001): Novel local search-based approaches.
- P4: Burke and Newall (2003): Enhancing timetable solutions with local search methods
- P5: Merlot et al. (2003): A hybrid algorithm.
- P6: White et al. (2001): Tabu search with longer-term memory

- P7: Burke et al. (2007): A graph-based hyper-heuristic.
- P8: Abdullah et al. (2006): A multi-start large neighbourhood search approach with local search methods.

Table 10
Results obtained from HBMO-ETP compared to other (non-population based) methods in the literature. Best penalty costs are in bold.

| Data sets | HBMO-ETP | | P1 | | P2 | | P3 | P4 |
|---|---|---|---|---|---|---|---|---|
| | Ave | Best | Ave | Best | Ave | Best | Best | Best |
| Car-f-92-I | 4.30 | **3.90** | 7.0 | 6.2 | 5.6 | 5.2 | 6.0 | 4.10 |
| Car-s-91-I | 4.86 | 4.79 | 8.4 | 7.1 | 6.5 | 6.2 | 6.6 | 4.65 |
| Ear-f-83-I | 36.43 | 34.69 | 40.9 | 36.4 | 46.7 | 45.7 | **29.3** | 37.05 |
| Hec-s-92-I | 10.84 | 10.66 | 15.0 | 10.8 | 12.6 | 12.4 | **9.2** | 11.54 |
| Kfu-s-93 | 13.41 | **13.00** | 18.8 | 14.0 | 19.5 | 18.0 | 13.8 | 13.90 |
| Lse-f-91 | 10.56 | 10.00 | 12.4 | 10.0 | 15.9 | 15.5 | **9.6** | 10.82 |
| Pur-s-93-I | 6.3 | 4.76 | - | - | - | - | **3.7** | - |
| Rye-s-93 | 11.90 | 10.97 | 8.7 | 7.3 | - | - | **6.8** | - |
| Sta-f-83-I | 159.67 | **157.04** | 167.1 | 161.5 | 166.8 | 160.8 | 158.2 | 168.73 |
| Tre-s-92 | 8.00 | **7.87** | 10.8 | 9.6 | 10.5 | 10.0 | 9.4 | 8.35 |
| Uta-s-92-I | 3.28 | **3.10** | 4.8 | 3.5 | 4.5 | 4.2 | 3.5 | 3.20 |
| Ute-s-92 | 26.98 | 25.94 | 30.8 | 25.8 | 31.3 | 29.0 | **24.4** | 25.83 |
| Yor-f-83-I | 36.77 | **36.15** | 45.6 | 41.7 | 42.1 | 41.0 | 36.2 | 37.28 |

Table 11
Results obtained from HBMO-ETP compared to other (non-population based) methods in the literature. Best penalty costs are in bold.

| Data sets | HBMO-ETP | | P4 | | P6 | | P7 | P8 |
|---|---|---|---|---|---|---|---|---|
| | Ave | Best | Ave | Best | Ave | Best | Best | Best |
| Car-f-92-I | 4.30 | **3.90** | 4.4 | 4.3 | 4.7 | 4.63 | 5.36 | 4.1 |
| Car-s-91-I | 4.86 | 4.79 | 5.2 | 5.1 | 5.8 | 5.73 | **4.53** | 4.8 |
| Ear-f-83-I | 36.43 | 34.69 | 35.4 | 35.1 | 46.4 | 45.8 | 37.92 | 36.0 |
| Hec-s-92-I | 10.84 | 10.66 | 10.7 | 10.6 | 13.4 | 12.9 | 12.25 | 10.8 |
| Kfu-s-93 | 13.41 | **13.00** | 14.0 | 13.5 | 17.8 | 17.1 | 15.2 | 15.2 |
| Lse-f-91 | 10.56 | 10.00 | 11.0 | 10.5 | 14.8 | 14.7 | 11.33 | 11.9 |
| Pur-s-93-I | 6.3 | 4.76 | - | - | - | - | - | - |
| Rye-s-93 | 11.90 | 10.97 | 8.7 | 8.4 | 11.7 | 11.6 | - | - |
| Sta-f-83-I | 159.67 | **157.04** | 157.4 | 157.3 | 158 | 158 | 158.19 | 159.0 |
| Tre-s-92 | 8.00 | **7.87** | 8.6 | 8.4 | 9.2 | 8.94 | 8.92 | 8.5 |
| Uta-s-92-I | 3.28 | **3.10** | 3.6 | 3.5 | 4.5 | 4.44 | 3.88 | 3.6 |
| Ute-s-92 | 26.98 | 25.94 | 25.2 | 25.1 | 29.1 | 29.0 | 28.01 | 26.0 |
| Yor-f-83-I | 36.77 | **36.15** | 37.9 | 37.4 | 49.5 | 42.3 | 41.37 | 36.2 |

We can see that our proposed HBMO-ETP has produced good quality solutions (with regard to our best and average results) for 7 instances (Car-f-92, Kfu-s-93, Pur-s-93, Sta-f-83, Tre-s-92, Uta-s-92, and Yor-f-83) when compared to those produced by other approaches in the literature. Indeed, we also obtained competitive results with other approaches in the literature for other instances.

### 5.3 The HBMO-ETP for the Socha Course Timetabling Benchmark Dataset

Table 12 presents the characteristics of the Socha benchmark dataset (Socha and Samples 2003). The benchmark consists of 11 problem instances which are categorized as small, medium and large.

Table 12
The Socha benchmark course timetabling dataset

| | Small | Medium | Large |
|---|---|---|---|
| Number of courses | 100 | 400 | 400 |
| Number of rooms | 5 | 10 | 10 |
| Number of timeslots | 45 | 45 | 45 |
| Number of features | 5 | 10 | 10 |
| Approx features per room | 3 | 3 | 5 |
| Percent feature use | 70 | 80 | 90 |
| Number of students | 80 | 200 | 400 |
| Max events per student | 20 | 20 | 20 |
| Max students per event | 20 | 50 | 100 |

In order to assess the performance of the proposed HBMO-ETP, we first compare it with the original HMBO (using equation (3) and without population updating strategy) using the same parameter settings as before (Table 5). Again, 20 runs were carried out for each of the 11 problem instances using different random seeds, each taking1-6 hours depending on the size and constraints in the instances (e.g. small 1 to small 5 takes a maximum 1 hour, whereas, medium and large takes 3 and 6 hours, respectively). Table 13 presents the best penalty cost (from the 20 runs) and average cost.

Table 13
Results obtained from HBMO-ETP compared to original one. Best penalty costs are in bold.

| Datasets | HBMO-ETP | | Original HBMO | |
|---|---|---|---|---|
| | Ave | Best | Ave | Best |
| small1 | 0 | **0** | 2 | 0 |
| small2 | 0 | **0** | 2 | 0 |
| small3 | 0 | **0** | 1 | 0 |
| small4 | 0 | **0** | 3 | 0 |
| small5 | 0 | **0** | 4 | 0 |
| medium1 | 79 | **75** | 101 | 98 |
| medium2 | 93 | **88** | 140 | 133 |
| medium3 | 134 | **129** | 217 | 201 |
| medium4 | 81 | **74** | 159 | 139 |
| medium5 | 73 | **64** | 190 | 178 |
| large | 531 | **523** | 840 | 827 |

It is clear from Table 13 that the best results obtained by HBMO-ETP outperform HBMO. Although the best results of the small instances are the same (both methods obtained zero cost), on average HBMO-ETP is more stable, in that the best and average results are the same (zero). Again, we believe this is due to the use of the population updating strategy to maintain diversity.

The performance of HBMO-ETP has been compared with five population based methods in Table 14 on the same dataset:

- M1: MAX-MIN Ant System by Socha and Samples (2003).
- M2: Genetic algorithm and local search by Abdullah and Turabieh (2008).
- M3: Hybrid evolutionary approach by Abdullah et al. (2007).
- M4: Evolutionary non-linear great deluge by Landa-Silva and Obit (2009).
- M5: Electromagnetism-like mechanism with force decay rate great deluge by Turabieh et al (2009)

It can be seen that across all problem instances, HBMO-ETP has produced much better results when compared against all the population based methods. For the five small instances, HBMO-ETP is able to

solve them to the optimality. We can also see that both the best and average results are better than the results obtained by other population based methods except the average for the large instance, which is slightly worse than that of M3.

Table 14
Results obtained from HBMO-ETP compared to the population based methods in the literature. Best penalty costs are in bold.

| Datasets | HBMO-ETP | | M1 | M2 | | M3 | | M4 | M5 |
|---|---|---|---|---|---|---|---|---|---|
| | Ave | Best | Best | Ave | Best | Ave | Best | Best | Best |
| small1 | 0 | **0** | 1 | 2.4 | 2 | 0 | **0** | 0 | **0** |
| small2 | 0 | **0** | 3 | 4 | 4 | 0 | **0** | 0 | **0** |
| small3 | 0 | **0** | 1 | 2.3 | 2 | 0 | **0** | 0 | **0** |
| small4 | 0 | **0** | 1 | 2 | 0 | 0 | **0** | 0 | **0** |
| small5 | 0 | **0** | 0 | 6.4 | 4 | 0 | **0** | 0 | **0** |
| medium1 | 79 | **75** | 195 | 275 | 254 | 224.8 | 221 | 126 | 175 |
| medium2 | 93 | **88** | 184 | 268 | 258 | 150.6 | 147 | 123 | 197 |
| medium3 | 134 | **129** | 248 | 262.8 | 251 | 252 | 246 | 185 | 216 |
| medium4 | 81 | **74** | 164 | 353 | 321 | 167.8 | 165 | 116 | 149 |
| medium5 | 73 | **64** | 219 | 284.4 | 276 | 135.4 | 130 | 129 | 190 |
| large | 531 | **523** | 851 | 1032.2 | 1027 | 552.4 | 529 | 821 | 912 |

Table 15 compares HBMO-ETP with other (non-population based) methods in the literature. These approaches are:

- S1: Graph hyper-heuristic by Burke et al. (2007).
- S2: Non-linear great deluge by Landa-Silva and Obit (2008).
- S3: Extended great deluge by McMullan (2007).
- S4: Variable neighbourhood search with tabu by Abdullah et al. (2005).
- S5: Great deluge and tabu search by Abdullah et al. (2009).

Table 15
Results obtained from HBMO-ETP compared to other (non-population based) methods in the literature. Best penalty costs are in bold.

| Datasets | HBMO-ETP | | S1 | S2 | S3 | | S4 | S5 | |
|---|---|---|---|---|---|---|---|---|---|
| | Ave | Best | Best | Best | Ave | Best | | Ave | Best |
| small1 | 0 | **0** | 6 | 3 | 0.8 | 0 | 0 | 0.8 | 0 |
| small2 | 0 | **0** | 7 | 4 | 2 | 0 | 0 | 2 | 0 |
| small3 | 0 | **0** | 3 | 6 | 1.3 | 0 | 0 | 1.4 | 0 |
| small4 | 0 | **0** | 3 | 6 | 1 | 0 | 0 | 1 | 0 |
| small5 | 0 | **0** | 4 | 0 | 0.2 | 0 | 0 | 0.6 | 0 |
| medium1 | 79 | **75** | 372 | 140 | 101.4 | 80 | 317 | 132.2 | 78 |
| medium2 | 93 | **88** | 419 | 130 | 116.9 | 105 | 313 | 124.6 | 92 |
| medium3 | 134 | **129** | 359 | 189 | 162.1 | 139 | 357 | 162 | 135 |
| medium4 | 81 | **74** | 348 | 112 | 108.8 | 88 | 245 | 111.2 | 75 |
| medium5 | 73 | **64** | 171 | 141 | 119.7 | 88 | 292 | 113.1 | 68 |
| large | 531 | **523** | 1068 | 876 | 834.1 | 730 | - | 738.6 | 556 |

Again, HBMO-ETP outperforms other approaches across all instances. Furthermore, both the best and average results are better than all other methods on all instances. These results demonstrate that HBMO-ETP is able to produce highly competitive results, which supports findings on the exam timetabling problems.

## 6 Conclusions

We have presented the first honey-bee mating optimization algorithm for solving educational timetabling problems (HBMO-ETP), which is based on the original HBMO algorithm.

HBMO mimics the mating flight of the queen bee. When the drones mate with her they deposit their sperm in her spermatheca, giving her many possible genotype combinations to generate new broods. One feature of HBMO is its ability to explore and exploit the search space simultaneously. The original HBMO suffers from premature convergence as the initial population is never updated or modified during the entire search process. The proposed HBMO-ETP maintains the population diversity by discarding mated drones and inserting the generated broods into the population for the next mating flight.

The original HBMO, and our proposed enhancement (HBMO-ETP), has been evaluated on both exam and course timetabling benchmark problems. Results show that HBMO-ETP produces highly competitive solutions for both of the benchmarks and outperforms the original HBMO. This is due to its ability to explore and exploit the search space. However, like other meta-heuristic algorithms, the main drawback of HBMO-ETP is the number of parameters that need to be set, such as the drone's population size, mating pool size and the brood's population size.

Now that we have demonstrated the effectiveness of this method on very well established benchmarks, we plan to widen our investigations to the new benchmark (exam and course) datasets that have recently been introduced (ITC 2007 datasets, McCollum et al. 2010), as well as other highly constrained optimization problems including nurse rostering.

## References


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, 2001a. Las Vegas, NV, USA.

Abbass, H.A., Marriage in honey-bee optimization (MBO): A haplometrosis polygynous swarming approach. CEC2001, 2001b, pp. 207-214. Seoul, Korea.

Abdullah, S. and Turabieh, H., Generating university course timetable using genetic algorithm and local search. Proceeding of the 3rd International Conference on Hybrid Information Technology. 2008, pp 254-260.

Abdullah, S., Burke, E. K. and McCollum, B., A hybrid evolutionary approach to the university course timetabling problem. CEC2007, 2007, ISBN: 1-4244-1340-0, pp 1764-1768.

Abdullah, S., Burke, E.K. and McCollum, B., An investigation of variable neighbourhood search for university course timetabling. The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA05), 2005, pp. 413-427.

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.

Abdullah, S., Shaker, K., McCollum, B., McMullan, P., Construction of Course Timetables Based on Great Deluge and Tabu Search: in MIC09, 2009, Germany.

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, 2007, 344, 452-462.

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.

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, December 2007, pp. 532, , Itech Education and Publishing, Vienna, Austria.

Burke, E.K., Eckersley, A.J., McCollum, B., Petrovic S., Qu, R., Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*, 2010, 206, 46-53.

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*, 2007, 176, 177–192.

Burke, E.K., Newall, J., Enhancing timetable solutions with local search methods. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, 2003, vol. 2740, pp. 195–206. Springer, Heidelberg.

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, 1996, vol. 1153, pp. 241-250. Springer, Heidelberg.

Burke, E.K., Petrovic, S., Recent research directions in automated timetabling. *European Journal of Operational Research*, 2002, 140, 266–280.

Caramia, M., Dell'Olmo, P., Italiano, G. F., New algorithms for examination timetabling. In: Naher, S., Wagner, D. (eds.) WAE 2000. LNCS, 2001, vol. 1982, pp. 230-241. Springer, Heidelberg.

Carter, M.W., Laporte, G., Lee, S.Y., Examination timetabling: Algorithmic strategies and applications. *Journal of Operational Research Society* 1996, 47(3), 373–383.

Costa, D., Hertz, A., Ant can colour graphs. *Journal of Operational Research Society*, 1997, 48, 295-305.

Cote, P. and Sabourin, R., A hybrid multi-objective evolutionary algorithm for the uncapacitated exam proximity problem. In: E.K. Burke and M. Trick (eds). Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling. Springer Lecture Notes in Computer Science, 2005, vol. 3616. 294-312.

Di Gaspero, L., Schaerf, A., Tabu search techniques for examination timetabling. In: Burke, E.K., Erben, W. (eds.) PATAT 2000. LNCS, 2001, vol. 2079, pp. 104–117. Springer, Heidelberg.

Eley, M., Ant algorithms for the exam timetabling problem. In: Burke, E.K., Rudova, H. (eds.) PATAT 2007. LNCS, 2007, vol. 3867, pp. 364-382. Springer, Heidelberg.

Ersoy, E., Ozcan, E. and Etaner., A.S., Memetic algorithms and hyper hill-climbers. In Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications. 2007, pp. 159-166.

Landa-Silva, D. and Obit, J.H., Great deluge with non-linear decay rate for solving course timetabling problem. The fourth international IEEE conference on Intelligent Systems. 2008, pp. 8.11–8.18, Varna, Bulgaria.

Landa-Silva, D. and Obit, J.H., Evolutionary Non-linear Great Deluge for University Course Timetabling: In proceeding of 2009 internation conference on hybrid artificial intelligent HAIS09, LNAI 5572, 2009, pp. 269–276 Springer-Verlag Berlin Heidelberg.

Lewis, R., A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 2008, 30(1), 167–190.

McCollum B., McMullan P., Paechter B., Lewis R., Schaerf A., Di Gaspero L., Parkes A., Qu R., Burke E., Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. INFORMS Journal on Computing. 2010, 22(1): 120-130.

McMullan, P., An extended implementation of the great deluge algorithm for course timetabling, Computational Science – ICCS, Part I, LNCS, 2007, 4487 pp 538–545 Springer-Verlag Berlin Heidelberg.

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, 2003, vol. 2740, pp. 207–231. Springer, Heidelberg.

Pillay, N. and Banzhaf, W., An informed genetic algorithm for the examination timetable problem. *Applied Soft Computing* 2010, 10 (2010) 457–467.

Qu, R., Burke, E.K., McCollum, B., Merlot, L.T.G., Lee, S.Y., A Survey of Search Approaches and Automated System Development for Examination Timetabling. *Journal of Scheduling*, 2009, 12(1), 55-89.

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), 2009a, pp. 90-94, Nashville, Tennessee, USA.

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. Roulette wheel graph colouring for solving examination timetabling problems. In Proceedings of the 3rd International Conference on combinatorial optimization and applications. LNCS 5573, (2009b). PP. 463 – 470. Berlin: Springer.

Socha, K. and Samples, M., Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art, in Evolutionary Computation in Combinatorial Optimization (EvoCOP 2003), 2003, vol. 2611, Lecture Notes in Computer Science pp. 334-345 Berlin Springer-Verlag.

Thompson, J.M., Dowsland, K.A., Variants of simulated annealing for the examination timetabling problem. *Annals of Operations research*, 1996, 63, 105-128.

Turabieh, H., Abdullah, S. and McCollum, B., Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem: in proceeding of The Fourth International Conference on Rough Set and Knowledge Technology, P. Wen et al. (Eds.): RSKT 2009, LNCS 5589, 2009, pp. 497–504, Springer-Verlag Berlin Heidelberg.

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, 2001, vol. 2079, pp. 85–103. Springer, Heidelberg.

Yang, X., *Nature Inspired Metaheuristic Algorithms*, Luniver Press, 2008.