Jurnal Teknologi

Full paper

A Modified Migrating Bird Optimization For University Course Timetabling Problem

Lam Way Shen[a], Hishammuddin Asmuni[a*], Fong Cheng Weng[b]

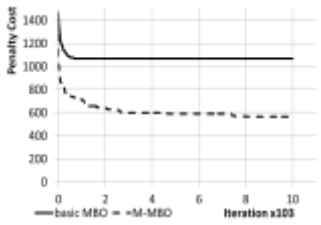
[a]*Soft Engineering Research Group (SERG), Faculty of Computing, Universiti Teknologi Malaysia, 81310 UTM Johor Bahru, Johor Malaysia*
[b]*Department of Computer Sciences and Mathematics, Faculty of Applied Sciences and Computing, Tunku Abdul Rahman University College, 85000 Segamat Johor, Malaysia*

[*]Corresponding author: hishamudin@utm.my

Article history

Received: 12 May 2014
Received in revised form:
1 September 2014
Accepted: 1 December 2014

Graphical abstract



Abstract

University course timetabling problem is a dilemma which educational institutions are facing due to various demands to be achieved in limited resources. Migrating bird optimization (MBO) algorithm is a new meta-heuristic algorithm which is inspired by flying formation of migrating birds. It has been applied successfully in tackling quadratic assignment problem and credit cards fraud detection problem. However, it was reported that MBO will get stuck in local optima easily. Therefore, a modified migrating bird optimization algorithm is proposed to solve post enrolment-based course timetabling. An improved neighbourhood sharing mechanism is used with the aim of escaping from local optima. Besides that, iterated local search is selected to be hybridized with the migrating bird optimization in order to further enhance its exploitation ability. The proposed method was tested using Socha's benchmark datasets. The experimental results show that the proposed method outperformed the basic MBO and it is capable of producing comparable results as compared with existing methods that have been presented in literature. Indeed, the proposed method is capable of addressing university course timetabling problem and promising results were obtained.

Keywords: Migrating bird optimization algorithm; iterated local search; neighbourhood sharing mechanism; post enrolment-based course timetabling

© 2014 Penerbit UTM Press. All rights reserved.

1.0 INTRODUCTION

Timetabling is a well-known difficult optimization problem and has been widely studied by many researchers since last two decades or earlier. Various types of timetabling problems have emerged in recent times, including university and school timetabling [1-4], nurse rostering [5-7], public transportation timetabling [8-10], tournament scheduling [11, 12] and television programs scheduling [13, 14]. In this paper, post-enrolment based university course timetabling is investigated.

The goal of university course timetabling problem (UCTP) is to allocate a number of courses to limited timeslots and rooms with minimum violation of desirable constraints [15]. Generally, there are two types of constraints: hard and soft constraints. Hard constraint must not be violated to ensure the feasibility of a timetable. A feasible timetable is known as clash-free timetable. On the other hand, soft constraint is an option to generate high quality timetable. Hence, violation on them should be minimized. In early stage, sequential heuristic methods are widely used to solve UCTP [16]. These methods demonstrate good performances in solving small instances problem, but they are not practicable for complicated problems, which is usually large and complex. In recent years, researchers turn their focus on meta-heuristic, hyper-heuristic, and hybridization methods. Several examples on these methods including Hill Climbing [4, 17, 18], Simulated Annealing [19, 20], Great Deluge [21, 22], Genetic Algorithm [3, 23], Particle Swarm Optimization [24, 25], and Harmony Search [26, 27]. In this paper, Migrating Birds Optimization (MBO) is utilized in solving university course timetabling problem.

The rest of the paper is organized as follows. The details of the university course timetabling problem are presented in Section 2. The basic concept of MBO algorithm and the proposed method are presented in Sections 3 and 4, respectively. Section 5 demonstrates and discusses on the experimental results. Finally, Section 5 concludes and suggests possible future work.

2.0 PROBLEM DESCRIPTION

Benchmark dataset that proposed by Socha et al. [28] is investigated in this paper. There are eleven problem instances in the datasets. The problem in the benchmark dataset consists of:

E, a set of courses ($E = \{e_1 \dots e_a\}$);
S, a set of students ($S = \{s_1 \dots s_b\}$);
T, a set of timeslots ($T = \{t_1 \dots t_c\}, where\ c = 45$);
R, a set of rooms ($R = \{r_1 \dots r_d\}$);
F, a set of features ($F = \{f_1 \dots f_n\}$);

The goal is to schedule courses E into limited timeslots T and rooms R and a way that satisfies a number of predefined constraints. Every room has a capacity limit and different

number of room features, every student attends a number of courses and each course required different number of room features. There are five small, five medium and one large problem in the dataset. Table 1 shows the detailed description of the problem.

Table 1 Socha course timetabling benchmark datasets

	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	3
Percent feature use	70	80	90
Number of students	80	200	400
Max courses per student	20	20	20
Max students per course	20	50	100

The problem consists of three hard constraints (HC) and three soft constraints (SC) as follows:

HC1: Student cannot be assigned to more than one course at the same timeslot.
HC2: The room capacity should not be less than the number of students attending the course assigned and the room should satisfy the features required by the course assigned.
HC3: Not more than one course should be assigned to each room in the same timeslot
SC1: Students should not have a single course in a day.
SC2: Students should not have more than two consecutive courses in a day.
SC3: Students should not have class in the last timeslot of the day.

■3.0 MIGRATING BIRD OPTIMIZATION ALGORITHM

This section describes the main idea of Migrating Bird Optimization (MBO) algorithm which was introduced by Duman et al. [29]. Generally, MBO algorithm imitates the behaviour of bird migration in V-shaped flight formation when season changes. There is a bird leading the flock, which is followed by other birds two lines both on the left and right side of the leader bird. Thus, a V shape formation is formed. When birds fly against air resistance or in free formation, they are challenged by huge induced power. Nevertheless, in the V-shaped formation, the leader bird uses the most energy to face induced power so that the remaining birds can save up to 20% energy [30, 31].The induced power is subsequently shared between the birds behind. When the leading bird is tired, it relocates to the end of the line; while the immediate next bird will take the lead. Usually, the strongest bird will lead the flock, and the leading bird is replaced cyclically until the final destination is reached.

In MBO algorithm, the birds (solutions in the population) fly in V-shaped formation (the solutions aligned in V-shaped form), leader bird face higher induced power (number of neighbourhood solutions generated) than the remaining birds in that formation as the induced power is shared between the birds behind (neighbourhood sharing mechanism). MBO describes how the migrating birds flock flying from one location to another with minimum energy used while in optimization, it is described as generating high quality solution with low energy spent. The energy here can be defined as the number of better quality neighbourhood solutions that can be found. Less energy left means few or no better neighbourhood solutions found while less energy spent means only few better neighbourhood solutions are used.

MBO algorithm starts with a number of initial solutions, one of the solutions is chosen as a leader solution and all of the solutions are placed on left line and right line to develop a V-shaped formation. Each solution in the formation attempts to be improved by generating neighbourhood solutions, starting from the leader solution and followed by other solutions on the lines until the end of the formation. If a better quality neighbourhood solution is found, then the current solution is replaced. Besides that, there is a neighbourhood sharing mechanism in-between current solution and the solutions that follows. The neighbourhood sharing mechanism is to share the best unused neighbourhood solutions to the next solution, the "unused" means a neighbourhood solution which has not been used in the solution replacement process by current solution. Therefore, except the leader solution is improved from its own neighbourhood solutions, the remaining solutions will get a number of best unused neighbourhood solutions shared in front of them. By combining with its own generated neighbourhood solutions, it is replaced by a better neighbourhood solution among them. The termination criterion of this procedure is based on the number of iterations (tours). After that, the leader solution is moved to the end of the line and one of the following solutions is forwarded following it is forwarded to the leader position. The leader solution replacement considers one line per replacement, and it turns around in left and right lines. Once the leader position is replaced, the next loop continues. The algorithm stops when termination criterion is met. Figure 1 shows the framework of basic MBO.

MBO has been studied and applied in tackling real world problems such as quadratic assignment problem [29], flow shop scheduling [32] and credit cards fraud detection problem [33]. In this paper, basic MBO has been applied in UCTP successfully. Table 3 shows the experimental results that were obtained. The result illustrates that, the performance of basic MBO in addressing UCTP is not very promising. This is due to basic MBO getting stuck in local optima easily as well as weak exploitation in search process. This can be related with the neighbourhood structure used in basic MBO. As reported by Burke et al. [34], the quality of solution generated and connectivity of search space is highly dependent on the neighbourhood structures. In basic MBO, only one neighbourhood structure is used to generate tentative solution, which is simple swap. In fact, it might restrict the search space and contribute to the trapping in local optima. Hence, when most of the solutions in a population converge to one point, chances of getting the better solutions become lower.

In the investigation carried out by Duman and Elikucuk [33], only one of the different neighbourhood structures is outperformed in MBO. This shows that, if an algorithm is applied with weak neighbourhood structure, the connectivity of the solutions search region is relatively weak and the search process will heavily rely on the initial solutions. Besides that, the use of single neighbourhood structure in the algorithm has lower chances of reaching global optima [35]. Therefore, Gao et al. [32] improve exploration ability of MBO by introducing multiple migrating bird swarm and several neighbourhood structures.

In addition, Duman et al. [29] reported that, MBO has best performance with 51 number of birds. In other words, MBO is

easily trapped in local optima with small population size. Moreover, another factor which causes the search process to be stuck easily in local optima is due to the fact that MBO utilizes greedy selection scheme in neighbourhood sharing mechanism. In this case, neighbourhood sharing mechanism accepts only better or equivalent quality solution and it causes the population to pre-maturely converge. Besides, the leader solution replacement in MBO utilizes static linear selection scheme, which position numbers are assigned to each solution in the initialization phase and the position remains unchanged until the search process ends. It affects the neighbourhood sharing mechanism and always shares neighbourhood solution to the same solution next to it. This imbalance sharing weakens the exploration abilities of the mechanism.

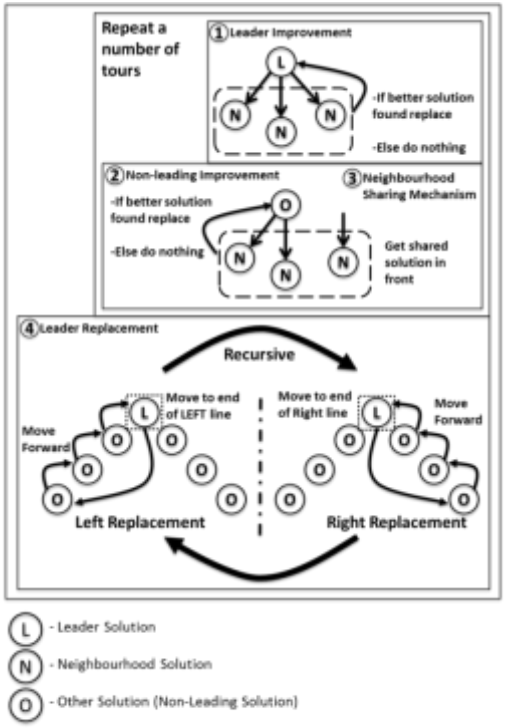


Figure 1 Framework of basic MBO

■4.0 MODIFEID MIGRATING BIRD OPTIMIZATION ALGORITHM (M-MBO)

This section describes the proposed method of the Modified Migrating Bird Optimization algorithm (M-MBO) in detail. The goals of this method are to avoid deadlock in the local optima and to enhance convergence speed in basic MBO. Furthermore, a variant of leader solution replacement in MBO was introduced to increase the performance. Figure 2 shows the framework of the proposed method. There are four differences between basic MBO and proposed method. Firstly, iterated local search was added to enhance the exploitation strength. Second, the repeating tour was eliminated to reduce the computational time. Third, neighbourhood sharing mechanism is introduced for the solution that failed to improve and forth, leader replacement was changed from recursive to random in between left and right lines. Generally, the proposed method is divided into two phases, which are: initialization and improvement. There are three main sections in improvement phase, which are: leader solution improvement, non-lead solution improvement, and leader solution replacement.

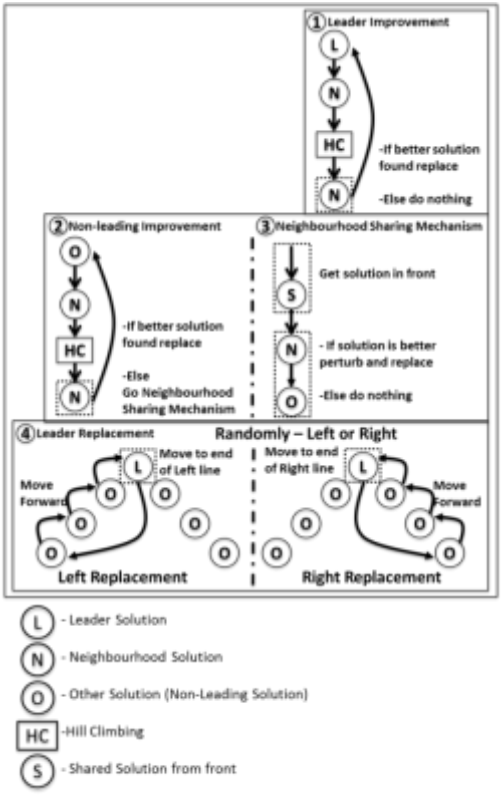


Figure 2 Framework of proposed method

4.1 Initialization Phase

In lines 4-9 (refer Figure 3), N numbers of solutions in the population are initialized and the penalty cost for each solution is calculated. All solutions in the population are feasible initial solutions. Initial solutions are generated by using combination of multiple graphs colouring heuristics, which includes: largest enrolment first, largest degree first and saturation degree. Each solution is assigned with a position number. Zero position number represents leader solution, odd position number represents left line and even position number represents right line. Figure 4 shows the visualization of V-shaped formation in the algorithm.

4.2 Improvement Phase

Lines 12-32 represent the improvement phase, which includes leader solution improvement, non-leading solution improvement and leader solution replacement. The entire process from basic MBO is simplified and some parts have been modified or eliminated. For example, the repeating tour in basic MBO was eliminated. The following sections describe the details of the improvement processes.

4.2.1 Leader Solution Improvement

In this section, the neighbourhood search process in basic MBO is replaced by Iterated Local Search (ILS) to increase convergence speed and escape from local optima. Figure 5 shows the detailed processes of ILS. There are two phases in ILS, neighbourhood move and hill climbing. In ILS, the selected solution generates a neighbourhood solution by performing neighbourhood move. In this paper, four different neighbourhood structures were employed, as follows:

NB1: Simple Move Event, which randomly selects an event and moves it to another feasible timeslot and room.
NB2: Simple Swap Event, which randomly selects two events and swaps their timeslot and room. Only feasible swap is accepted.
NB3: Simple Swap Timeslot, which randomly selects two timeslots and swaps the events which occupied in the timeslots.
NB4: Multiple Move Event, which randomly select two or more, up to ten events and move them to another feasible timeslots and rooms.

```
1.  Modified Migrating Bird Optimization
    Algorithm (M-MBO)
2.
3.  Initialization:
4.  Initialize population size, N;
5.  //population size = number of birds = number
    of birds in flock
6.  Initialize maximum number of iteration, I;
7.  Initiate population with feasible initiate
    solution;
8.  Calculate objective penalty cost for each
    solution, f(Sol)
9.  Assign position number to each solution;
10.
11. Improvement:
12. For i = 1 to I
13.    //Leader Solution Improvement Stage
14.    Try to improve leader solution by using
       Iterated Local Search
15.
16.    //Non-leading Solution Improvement Stage
17.    For j = 1 to N
18.        Try to improve non-leading solution,
           Solj by using Iterated Local Search
19.        If (Solj is not improved)
20.            Performs Neighbourhood Sharing
               Mechanism
21.        End If
22.    End For
23.
24.    //Leader Solution Replacement Stage
25.    Move leader solution to the end
26.    Replace leader position with one of the
       following solutions
27. End For
```

Figure 3 Pseudo-code of M-MBO

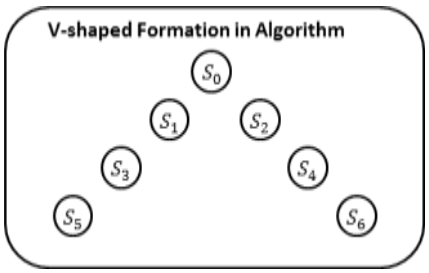


Figure 4 V-shaped formation in algorithm

In addition, according to the migrating bird's story in [29], the leader solution faces more induced power while the non-leading solutions, which are divided into left and right lines take less induced power.and the non-leading solution is divided into left line and right line take less induced power. Two neighbourhood structures are used in describing the situation that the leader solution faces as regards acquiring more induced power, which applied NB2 and continues with NB3. Besides, left and right lines should face induced power from different direction, NB2 and NB4 respectively. Regardless on the fitness value evaluation, the neighbourhood solution is generated directly and undergoes hill climbing improvement by using NB1.

Compared with basic ILS, ILS in M-MBO eliminates the local search in initialization of basic ILS. The main reason is to start hill climbing searching from neighbourhood position of a selected solution, even the solution is not trapped in local optima.

```
1.  Iterated Local Search
2.
3.  Initialization:
4.  Initialize number of iteration for Hill
    Climbing, K
5.
6.  Improvement:
7.  //Neighbourhood Move
8.  Generate a neighbourhood solution, Solj* by
    using NB2, NB3, or NB4
9.
10. //Hill Climbing:
11. For k = 1 to K
12.    Select Solj* and generate Solj** by using
       NB1
13.    If  f(Solj**) <= f(Solj*)
14.        Solj* = Solj**
15.    End If
16. End For
```

Figure 5 Pseudo-code of Iterated Local Search

4.2.2 Non-Leading Solution Improvement

In this section, ILS is utilized to replace neighbourhood search improvement as stated in section 2.1. Besides, since sharing neighbourhood solutions in each iteration has been defined, neighbourhood sharing mechanism is applied only when ILS is unable to improve the current solution. The section below describes the process of neighbourhood sharing mechanism.

4.2.2.1 Neighbourhood Sharing Mechanism

Figure 6 represents the pseudo-code of neighbourhood sharing mechanism. Generally, neighbourhood sharing mechanism share solutions to the solution next to it. As can be seen in Figure 7, only leader solution shares solutions to the first solution in left and right lines. Starting from the first solution in the left and right lines, the solution is shared to the next solution in the same line only. Hence, the left and right lines are independent from each other, which means there is no communication in any form between the two lines.

Since there is no neighbourhood solution generated when improving all solutions, in neighbourhood sharing mechanism share the solution itself. However, the solution next to it, accepts only shared solution if the shared solution has better quality. Despite being replaced with the same solution, the shared solution has a small perturbation by using NB4 to explore other searching regions.

4.2.3 Leader Solution Replacement

In this stage, the leader solution is moved to end of line and followed solution is moved forward.. The leader solution replacement considers one line per replacement. Figure 8 demonstrates the replacement move. Static linear selection

scheme here is replaced by a random selection scheme, which allocates 50 percent to the left line and 50 percent to the right line.

```
1.  Neighbourhood Sharing Mechanism
2.
3.  Get the solution in front of Solj, Soljf
4.  If f(Soljf) < f(Solj)
5.     Solj = Soljf
6.     Solj perform perturbation by using NB4
7.  End If
```

Figure 6 Pseudo-code of Neighbourhood Sharing Mechanism

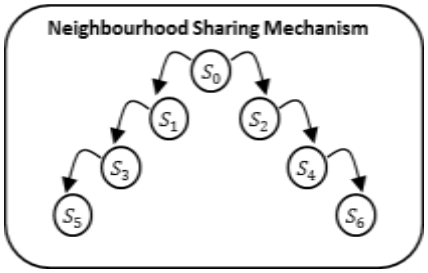


Figure 7 Neighbourhood Sharing Mechanism

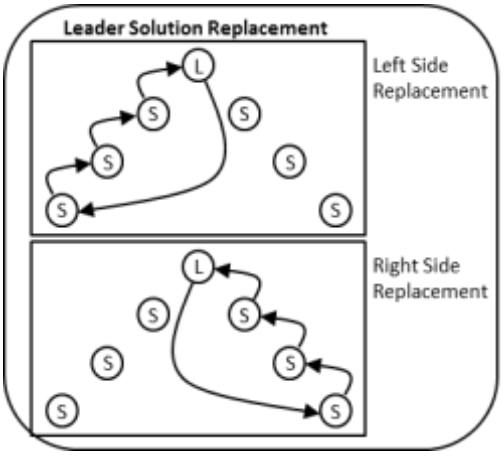


Figure 8 Leader Solution Replacement

■5.0 COMPUTATIONAL RESULT

The proposed method was coded with C++ programming language, and the experiments were conducted on a laptop with an Intel Pentium B960 2.2GHz, 4GB RAM, Windows 7 Ultimate. The proposed method was evaluated with respect to addressing course timetabling problem which was discussed in Section 2 with 10 times across 11 instances and each running at maximum of 10,000 iterations. Table 2 presents the parameter setting for M-MBO and basic MBO. The computational time consumed for instance was between 40 to 300 seconds and 40 to 3328 seconds for basic MBO and M-MBO, respectively. Table 3 illustrates the comparison results between the basic MBO, proposed method and best known results in the literature. The best results are highlighted in bold font. The generated results were validated using validator program which is provided in http://www.cs.qub.ac.uk/itc2007/postenrolcourse/course_post_index_files/validation.htm. As shown in Table 3, M-MBO outpaced the basic MBO in all instances. Also, M-MBO is able to obtain optimal solutions for all small instances. However, the results for medium and large instances were unable to compete with the current best known result (except for medium 05).

Table 4 illustrates the results comparison of the methods proposed in recent years and the best results are highlighted in bold font. The selected methods include:

M1: Hybrid Great Deluge with Tabu Search by Shaker et al. [36]
M2: Modified Artificial Bee Colony by Bolaji et al. [37]
M3: Hybrid Harmony Search algorithm by Al-Betar et al. [27]
M4: Hybrid Genetic Algorithm by Karami and Hasanzadeh [38]
M5: Population based Local Search by Abuhamdah et al. [39]
M6: Scatter Search by Jaradat et al. [40]
M7: Big Bang-Big Crunch by Jaradat and Ayob [41]
M8: Electromagnetic-like Great Deluge by Abdullah et al. [42]
M9: Honey-bee Mating Optimization by Sabar et al. [43]

Table 2 Parameter setting used for M-MBO and Basic MBO

Parameter	M-MBO	Basic MBO
Number of solutions in population, *N*	11	51
Number of neighbourhood solutions generated, *n*	-	3
Number of neighbourhood solutions shared, *x*	-	1
Number of tour, *J*	-	10
Maximum iteration, *I*	10,000	10,000
Number of iteration for Hill Climbing, *K*	1,000	-

From Table 4, it can be seen that M-MBO was unable to produce new best results. However, it still managed to produce results that were ranked third place for medium 03 and medium 05 instances. In addition, M-MBO is capable of generating good quality result for medium 05 and large instances which were classified as the most difficult instance [37].

Figure 9 represents the convergence graph for M-MBO and basic MBO in solving large instances. The x-axis represents the number of iteration and y-axis represents penalty cost value. It can be observed that the slope for M-MBO achieves much lower compared with the basic MBO, which indicates presence of a great improvement of the solution quality. As can be seen at 1000 iterations, the penalty cost for basic MBO and M-MBO are about 1060 and 710, respectively. The improvement in solution quality is more than 30 percent. Furthermore, it can be observed that, the basic MBO was trapped in local optima after 1000 iterations of execution. In contrast, there is still enhancement in the solution quality for M-MBO even though the enhancement is slower than the beginning of the search. There is no enhancement of the solution quality after 8000 iterations. It is believed that, the search was trapped in a particular search region and was unable to explore other un-visited search spaces which might have better quality solutions. Neighbourhood sharing mechanism in M-MBO might be the culprit which trapped the search in local optima. In this mechanism a solution shares itself and replaces the solution next to it if it has better quality. When a dominant super-individual exists, the whole population tends to converge towards its direction. This will results the population trapped in local optima if no better quality of solution is obtained.

Figure 10 shows the box plot of penalty cost for all instances. The gap between best, average and worst penalty costs for all small instances are zero. For medium and large instances except mediums 02 and 03, the average penalty costs are closer to the best than the worst. This indicates that M-MBO is stable and mostly able to produce good quality solution.

6.0 CONCLUSION

In this paper, a variance of basic MBO algorithm, M-MBO was presented to solve UCTP. The simple neighbourhood search process was replaced by ILS to improve the exploitation ability. An improved neighbourhood sharing mechanism was introduced in order to avoid the search process from converging towards one search area easily. The basic MBO and M-MBO were tested on eleven problem instances from Socha's benchmark dataset. Experimental results show that M-MBO outpaced the basic MBO. It is believed that hybridization of ILS and the improved neighbourhood sharing mechanism in M-MBO bring a great improvement in quality as it possesses strong exploitation ability and capable of escaping from local optima compared with Basic MBO. However, the exploration ability of M-MBO is weak, so there is still room for improvement. The exploration ability of the M-MBO can be further enhanced by incorporating with other exploration method and this is subject to future work.

Table 3 Results comparison on basic MBO and M-MBO

Dataset	MBO		M-MBO		Best Known	
	Best	Average	Best	Average		
Small01	25	30.6	**0**	**0**	**0**	many
Small02	22	28.4	**0**	**0**	**0**	many
Small03	19	23.8	**0**	**0**	**0**	many
Small04	14	19.9	**0**	**0**	**0**	many
Small05	17	34.7	**0**	**0**	**0**	many
Medium01	394	418.8	115	132	**41**	Abuhamdah et al. [39]
Medium02	378	425.2	120	137.8	**39**	Abuhamdah et al. [39]
Medium03	305	339	124	136.6	**60**	Abuhamdah et al. [39]
Medium04	383	411.8	110	124.8	**39**	Abuhamdah et al. [39]
Medium05	276	292.8	61	97.8	**53**	Al-Betar et al. [27]
Large	1015	1049.2	553	583.9	**385**	Al-Betar et al. [27]

Table 4 Results comparison with method proposed in literature

Dataset	M-MBO Best	M1 Best	M2 Best	M3 Best	M4 Best	M5 Best	M6 Best	M7 Best	M8 Best	M9 Best
Small01	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small02	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small03	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small04	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Small05	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Medium01	115	78	129	99	180	**41**	70	99	96	75
Medium02	120	92	119	73	176	**39**	77	102	96	88
Medium03	124	135	137	130	219	**60**	115	158	135	129
Medium04	110	75	146	105	150	**39**	67	86	79	74
Medium05	61	68	63	**53**	196	55	64	79	87	64
Large	553	556	525	**385**	-	463	555	768	683	523

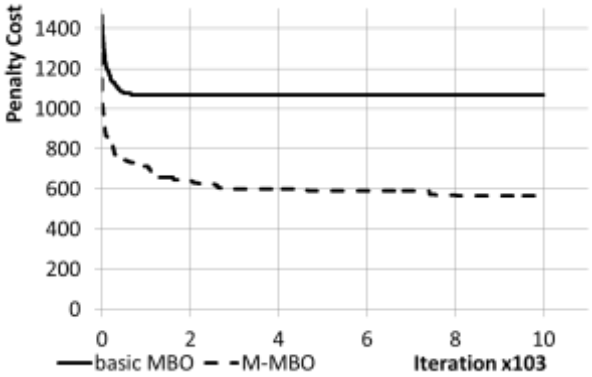


Figure 9 Convergence graph of M-MBO and basic MBO for large instance

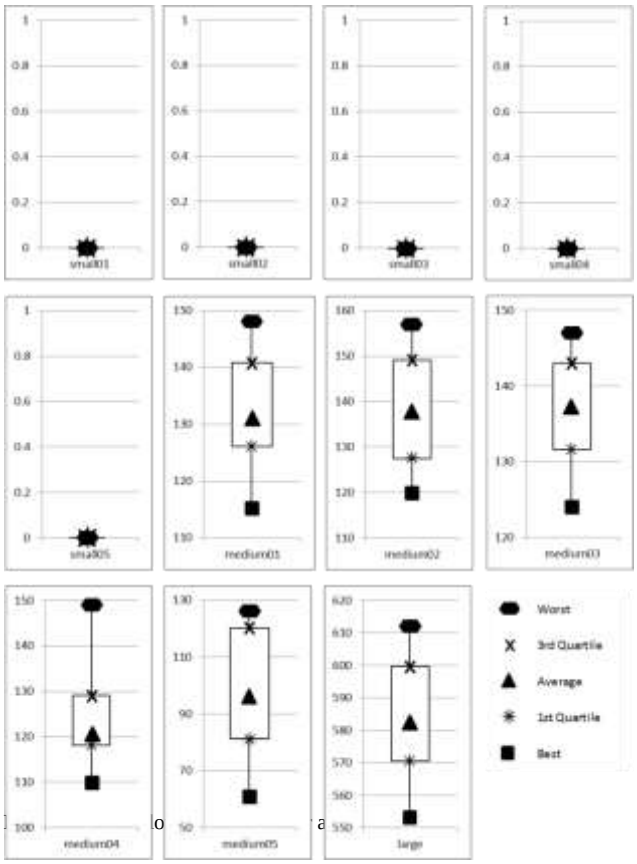


Figure 10 Box plots of M-MBO for all instances

Acknowledgement

The authors would like to acknowledge Zamalah Scholarship provided by Universiti Teknologi Malaysia and the Ministry of Higher Education Malaysia..

References

[1] Colorni, A., M. Dorigo, and V. Maniezzo. 1998. Metaheuristics for High School Timetabling. *Computational Optimization and Applications*. 9(3): 275–298.

[2] Beligiannis, G.N., C.N. Moschopoulos, G.P. Kaperonis, and S.D. Likothanassis. 2008. Applying evolutionary computation to the school timetabling problem: The Greek case. *Computers & Operations Research*. 35(4): 1265–1280.

[3] Burke, E.K., D.G. Elliman, and R.F. Weare. 1994. A Genetic Algorithm based University Timetabling System. *Proceedings of the 2nd East-West International Conference on Computer Technologies in Education*. 19th-23rd Sept. Crimea, Ukraine.35–40.

[4] Burke, E.K., J.P. Newall, and R.F. Weare. 1996. A memetic algorithm for university exam timetabling. *Practice and Theory of Automated Timetabling*. E. Burke and P. Ross, Springer Berlin Heidelberg. 1153: 241–250.

[5] Burke, E.K., G. Kendall, and E. Soubeiga. 2003. A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics* 9(6): 451–470.

[6] Özcan, E. 2005. Memetic Algorithms for Nurse Rostering. *Computer and Information Sciences - ISCIS 2005*. Yolum, et al., Springer Berlin Heidelberg. 3733**:** 482–492.

[7] Burke, E.K., T. Curtois, G. Post, R. Qu, and B. Veltman. 2008. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research* 188(2): 330–341.

[8] Caprara, A., M. Fischetti, P.L. Guida, M. Monaci, G. Sacco, and P. Toth. 2001. Solution of real-world train timetabling problems. *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*. 3-6 Jan. 10

[9] Caprara, A., M. Fischetti, and P. Toth. 2002. Modeling and Solving the Train Timetabling Problem. *Operations Research* 50(5): 851–861.

[10] Liu, Z.-g. and J.-s. Shen. 2007. Regional Bus Operation Bi-level Programming Model Integrating Timetabling and Vehicle Scheduling. *Systems Engineering - Theory & Practice* 27(11): 135–141.

[11] Schaerf, A. 1999. Scheduling Sport Tournaments using Constraint Logic Programming. *Constraints* . 4(1): 43–65.

[12] Rasmussen, R.V. and M.A. Trick. 2008. Round robin scheduling – a survey. *European Journal of Operational Research*. 188(3): 617–636.

[13] Rust, R.T. and N.V. Eechambadi. 1989. Scheduling Network Television Programs: A Heuristic Audience Flow Approach to Maximizing Audience Share. *Journal of Advertising*. 18(2): 11–18.

[14] Wuang, M.S., C.L. Yang, R.H. Huang, and S.P. Chuang. 2010. Scheduling of television commercials. *Industrial Engineering and Engineering Management (IEEM). 2010 IEEE International Conference on*. 7-10 Dec. 2010.803–807.

[15] Carter, M., E. Burke , M. Carter and G. Laporte. 1998. Recent developments in practical course timetabling. *Practice and Theory of Automated Timetabling II*. Springer Berlin Heidelberg. 1408: 3–19.

[16] de Werra, D. 1985. An introduction to timetabling. *European Journal of Operational Research* 19(2): 151–162.

[17] Merlot, L.G., N. Boland, B. Hughes, E. Burke. P. Causmaecker, Springer Berlin Heidelberg and P. Stuckey. 2003. A Hybrid Algorithm for the Examination Timetabling Problem. *Practice and Theory of Automated Timetabling IV*. 2740: 207–231.

[18] Ahandani, M.A. and M.T. Vakil-Baghmisheh. 2011. Examination timetabling using a hill climbing with combined neighbourhood structure. *Computer and Knowledge Engineering (ICCKE). 2011 1st International eConference on*. 13-14 Oct. 2011. 45–48.

[19] Kostuch, P. 2003. Timetabling competition-sa-based heuristic.

[20] Alzaqebah, M. Wang, X. Zhu, D.-Z. Du, Springer Berlin Heidelberg.and S. Abdullah. 2011. Hybrid Artificial Bee Colony Search Algorithm Based on Disruptive Selection for Examination Timetabling Problems. *Combinatorial Optimization and Applications*. 6831: 31–45.

[21] Burke, E., Y. Bykov, J. Newall, and S. Petrovic. 2003. A time-predefined approach to course timetabling. *The Yugoslav Journal of Operations Research ISSN: 0354-0243 EISSN: 2334-6043* 13(2).

[22] Obit, J. V. Sgurev, M. Hadjiski, J. Kacprzyk, Springer Berlin Heidelberg.and D. Landa-Silva. 2010. Computational Study of Non-linear Great Deluge for University Course Timetabling. *Intelligent Systems: From Theory to Practice*. 299: 309–328.

[23] Jat, S. and S. Yang. 2011. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. *Journal of Scheduling* 14(6): 617–637.

[24] Li, L. and S.-h. Liu. 2011. Study of course scheduling based on particle swarm optimization. *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2011*. 26-30 July 2011.1692–1695.

[25] Chen, R.-M. and H.-F. Shih. 2013. Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search. *Algorithms*. 6(2): 227–244.

[26] Al-Betar, M. and A. Khader. 2012. A harmony search algorithm for university course timetabling. *Annals of Operations Research*. 194(1): 3–31.

[27] Al-Betar, M.A., A.T. Khader, and M. Zaman. 2012. University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions*. 42(5): 664–681.

[28] Socha, K., J. Knowles, M. Dorigo, G. Caro, Springer Berlin Heidelberg. and M. Sampels. 2002. A MAX-MIN Ant System for the University Course Timetabling Problem. *Ant Algorithms*. 2463: 1–13.

[29] Duman, E., M. Uysal, and A.F. Alkaya. 2012. Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences*. 217: 65–77.
[30] Portugal, S.J., T.Y. Hubel, J. Fritz, S. Heese, D. Trobe, B. Voelkl, S. Hailes, A.M. Wilson, and J.R. Usherwood. 2014. Upwash exploitation and downwash avoidance by flap phasing in ibis formation flight. *Nature*. 505(7483): 399–402.
[31] Maeng, J.-S., J.-H. Park, S.-M. Jang, and S.-Y. Han. 2013. A modeling approach to energy savings of flying Canada geese using computational fluid dynamics. *Journal of Theoretical Biology*. 320(0): 76–85.
[32] Gao, K.Z., P.N. Suganthan, and T.J. Chua. 2013. An enhanced migrating birds optimization algorithm for no-wait flow shop scheduling problem. *Computational Intelligence in Scheduling (SCIS), 2013 IEEE Symposium*. 16-19 April 2013. 9–13.
[33] Duman, E. and I. Elikucuk. Applying Migrating Birds Optimization to Credit Card Fraud Detection. *Trends and Applications in Knowledge Discovery and Data Mining*.2013. J. Li, et al., Springer Berlin Heidelberg. 7867**:** 416–427.
[34] Burke, E.K., A.J. Eckersley, B. McCollum, S. Petrovic, and R. Qu. 2010. Hybrid variable neighbourhood approaches to university exam timetabling. *European Journal of Operational Research*. 206(1): 46–53.
[35] Mladenović, N., D. Urošević, S.d. Hanafi, and A. Ilić. 2012. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. *European Journal of Operational Research*. 220(1): 270–285.
[36] Shaker, K., S. Abdullah, A. Alqudsi, and H. Jalab. 2013. Hybridizing Meta-heuristics Approaches for Solving University Course Timetabling Problems. *Rough Sets and Knowledge Technology*. P. Lingras, et al., Springer Berlin Heidelberg. 8171: 374–384.
[37] Bolaji, A.A., A. Khader, M. Al-Betar, M. Awadallah. Y. Tan, Y. Shi, and H. Mo, A Modified Artificial Bee Colony Algorithm for Post-enrolment Course Timetabling. *Advances in Swarm Intelligence*.2013. Springer Berlin Heidelberg 7928: 377–386.
[38] Karami, A.H. and M. Hasanzadeh. 2012. University course timetabling using a new hybrid genetic algorithm. *Computer and Knowledge Engineering (ICCKE), 2012 2nd International eConference on*. 18-19 Oct. 2012. 144–149.
[39] Abuhamdah, A., M. Ayob, G. Kendall, and N. Sabar. 2014. Population based Local Search for university course timetabling problems. *Applied Intelligence*. 40(1): 44–53.
[40] Jaradat, G., M. Ayob, and Z. Ahmad. 2012. On the performance of Scatter Search for post-enrolment course timetabling problems. *Journal of Combinatorial Optimization*. 1–23.
[41] Jaradat, G.M. and M. Ayob. 2013. Effect of Elite Pool and Euclidean Distance in Big Bang-Big Crunch Metaheuristic for Post-Enrolment Course Timetabling Problems. *International Journal of Soft Computing* 8(2): 96–107.
[42] Abdullah, S., H. Turabieh, B. McCollum, and P. McMullan. 2012. A hybrid metaheuristic approach to the university course timetabling problem. *Journal of Heuristics* 18(1): 1–23.
[43] Sabar, N.R., M. Ayob, G. Kendall, and R. Qu. 2012. A honey-bee mating optimization algorithm for educational timetabling problems. *European Journal of Operational Research*. 216(3): 533–543.