An improved Artificial Bee Colony for Course Timetabling

Asaju La'aro Bolaji[1,3], Ahamad Tajudin Khader[1], Mohammed Azmi Al-Betar[1,2], Mohammed A. Awadallah[1]
[1] School of Computer Sciences, Universiti Sains Malaysia (USM), 11800, Pulau Pinang, Malaysia.
[2] Department of Computer Science, Al-zaytoonah University, Amman-Jordan.
[3] Department of Computer Science, University of Ilorin, Ilorin, Nigeria.
Emails: {abl10_sa0739@student, tajudin@cs, mohbetar@cs, mama10_com018@student}.usm.my

***Abstract*—The Artificial Bee Colony Algorithm (ABC) is an emerging nature-inspired, metaheuristic optimisation algorithm. In this paper, an improved ABC algorithm is proposed for tackling Curriculum-Based Course Timetabling Problem (CB-CTT). The ABC as a population-based algorithm, the initial population is generated using Saturation Degree (SD) followed by Backtracking Algorithm (BA) to ensure that all the solutions in the population are feasible. The improvement loop in ABC used neighbourhood structures severally within the employed and onlooker bees operators in order to navigate the CB-CTT search space tightly. The performance of ABC is tested using dataset prepared by second international timetabling competition (ITC-2007), the ABC is able to achieved good quality results, yet these are not comparable with the best results obtained by other methods. Future work can be directed further improve the ABC operators to achieve a better results.**

Keywords: Curriculum Based Course Timetabling, Artificial Bee Colony Algorithm, Nature Inspired Algorithm, Neighbourhood structure

I. Introduction

Nature inspired metaheuristic algorithms, such as Artificial Immune System (AIS) [1], Ant Colony Optimisation (ACO) [2], Firefly Algorithm (FA) [3], Genetic Algorithm (GA) [4], harmony search algorithm [5] and Particle Swarm Optimisation [6], have been applied by researchers to solve a wide range of timetabling and optimisation problems. Similarly, the Artificial Bee Colony Algorithm (ABC) introduced by Karaboga 2005 [7] was also recently tailored for the Curriculum Based Course Timetabling (CB-CTT) [8]. However, like other stochastic algorithms such as harmony search (HSA), particle swarm optimisation (PSO) and genetic algorithm (GA), as the number of solutions in the search spaces increases, ABC exhibits a poor convergence behaviour as shown in [8], [9]. Neighbourhood search can be the solution to this problem, which has been extensively studied in last four decades in the timetabling domain.

University Course Timetabling Problem (UCTP) is a challenging administrative process, which requiring to be repeated every semester in academic institutions such as Universities and Colleges. UCTP is referred to as a hard combinatorial optimisation problem [10], which was tackled using different techniques by workers in the fields of operational research and artificial intelligence. UCTP could be defined as the assignment of a set of courses to given periods and rooms, based on some given (hard and soft) constraints. The solution is said to be feasible, when it satisfies all hard constraints. UCTP could be split into two formations; Post enrolment course timetabling(PE-CTT) and Curriculum-based course timetabling problems (CB-CTT). The formulation 1 of CB-CTT problem was first proposed in [11] and a later version was introduced in 2007 as one of the tracks by the organisers of international timetabling competition (ITC-2007)[1] . The purpose of the competition was to close the gap existing between research and practice in the university timetabling domain [12]. Several optimisation techniques and algorithms have been applied to solve CB-CTT, where some of the algorithms developed include heuristics such as integer programming. [13], [14] and metaheuristics such as local search based and hybrid algorithms [4], [15], [16], [17], [18], [19].

Mathematical programming such as integer programming approach was proposed by Lach and lübbecke [13], where the method used was broken down into two steps: the initial step was used to guarantee the feasibility while the later step was to enhance the feasible solutions to the state-of-the-art results. The authors argued that the method was very robust because it steadily gave satisfactory lower and upper bound solutions, within a reasonable computation time without particular parameter tuning. unfortunately, the method still did not emerge as the overall winner in the competition. Burke et al., [14] presented branch and cut technique for CB-CTT problem; which was divided into two phases. An alternative integer programming formulation was employed at the first phase, where it used lower number of variables and slightly increased the number of soft constraints. In the second phase, the branch and cut procedure was used for further improvement. At this phase, the constraints necessary to attain optimality from enumeration of events/free-period patterns were also added. The results generated showed that it was possible to obtain provably optimal solutions for two instances within a reasonable time period by using the method. CB-CTT was also studied using hybrid solution algorithm called Adaptive Tabu Search (ATS)[20]. ATS technique was integrated with neighborhood structures and combined with perturbation from iterated local search. The results showed

[1]http://www.cs.qub.ac.uk/itc2007

that it results enhanced the previously reported . The ITC-2007 problem was investigated by Müller [21] and was evaluated as the best in the CB-CTT competition track. The method used one-stage solution approach which combined great deluge and simulated annealing incorporated into the constraint solver library. The iterative local search was used at the initial stage and the solution was further enhanced with the aid of constraint-based statistics (CBS).

In another development, the application of local search based approach for CB-CTT was presented in [15], where heuristics similar to squeaky wheel optimisation was used to obtain a feasible timetabling solution. The solution was further enhanced with the aid of threshold accepting criteria from simulated annealing, where the computation results showed that the algorithm was good for the problem. In another research, the CB-CTT was considered as a multi-objective optimisation problem [16]. The author presented a solution framework based on local search heuristics, using two different aggregation techniques i.e. a weighted sum aggregation and a reference point-based method. The experimental results showed that the approach was able to obtain good solutions. In a similar manner, the performance of the original ABC was investigated on CB-CTT [8], where the ABC run through two dependable phases; an initial phase using saturation degree (SD) to ensure feasible timetabling solution and the basic ABC used in the second phase to enhance the solution [8].

Unfortunately, the performance of the original ABC algorithm was unable to produce a result within the range obtained for previous methods [8]. To overcome this, an improved ABC algorithm is proposed in the present paper, by adding of two neighbourhood structures to the original ABC algorithm, to enhance the quality of the solutions.

The main objective of this paper is to presents an improvement for the ABC when applied to CB-CTT. The improvement includes incorporating two neighbourhood search mechanisms within the operators of ABC. For evaluating purpose, a dataset established by ITC-2007 was used. The results show that IABC can produce a fruitful results than basic ABC, yet not in the range of the state-of-the-art.

The rest of this paper is organised as follows; section II provides the detailed description and mathematical formulations of CB-CTT problem; the fundamentals of ABC and its proposed improvement using two basic neighbourhood structures is discussed in section III; Section IV provides the details of the computational results and the comparison with the average of result for the top five competitors. The final section presents the conclusion and possible future directions.

II. The Curriculum-Based Course Timetabling

The CB-CTT problem formulations and descriptions have been discussed in [22], where pertinent issues and detailed overview of CB-CTT were presented with respect to ITC-2007.

A. Problem Description

The CB-CTT is the scheduling of a set of lectures of courses to a set of rooms and periods on a weekly basis, in accordance with a given set of constraints [22]. Once all lectures of courses have been assigned to periods and rooms with respect to the hard constraints $(H_1 - H_4)$, then the timetable solution is said to be feasible. In addition, a feasible timetabling solution satisfying all hard constraints gives a penalty cost for the violations of the four soft constraints $(S_1 - S_4)$. The main objective is to minimise the number of soft constraint violations in a feasible solution. The four hard constraints and four soft constraints are outlined as follows:

Hard Constraints

- **H1 Lectures**: All lectures of a course must be assigned to a distinct period and a room.
- **H2 Room Occupancy**: Two lectures cannot be scheduled to the same room during the same period.
- **H3 Conflicts**: Lectures of courses in the same curriculum or taught by the same teacher must be scheduled to different periods.
- **H4 Availability**: If the teacher of a course is not available at a given period, then no lectures of the course can be allocated to that period.

Soft Constraints

- **S1 Room Capacity:** The number of students attending the course for each lecture must be less than or equal to the capacity of the rooms hosting the lectures
- **S2 Room Stability:** All lectures of a particular course should be assigned to the same room; otherwise, the number of occupied rooms should be less.
- **S3 Curriculum Compactness:** Lectures of courses belonging to the same curriculum should be in consecutive periods (i.e., adjacent to each other).
- **S4 Minimum Working Days:** The lectures of each course should be spread across a given number of days.

B. Problem formulation

The CB-CTT is the problem of assigning a set of D courses $\mathfrak{C} = \{c_1, c_2, \cdots, c_D\}$, to a set of H rooms $\mathfrak{R} = \{r_1, r_2, \cdots, r_H\}$ and a set of N periods $\mathfrak{P} = \{p_1, p_2 \cdots, p_N\}$ and a period is the composed of minimum working days M and timeslots per working days T (i.e. $\mathfrak{P} = M \times T$). Each course c_i consists of a set of lectures b_i which must be assigned to different periods. The problem consists of a set of Q curricula $\mathfrak{F} = \{f_1, f_2, \cdots, f_Q\}$ where each curriculum f_j is a group of courses having common students and finally, the CB-CTT consists of a set of R students $\mathfrak{S} = \{s_1, s_2, \cdots, s_R\}$, and each student is assigned with a set of courses within the same curricula.

Table 1. The symbols used for CB-CTT problem.

Symbols	Definition
D	The total number of courses
H	The total number of rooms
M	The total number of minimum working days per week
T	The total number of timeslots per days
N	The total number of periods, $P = M \times T$
Q	The total number of curricula
R	The total number of student

Symbols	Definition
$\mathfrak{C}$	Set of the courses, $\mathfrak{C} = \{c_1, ..., c_D\}$
$\mathfrak{R}$	Set of the rooms, $\mathfrak{R} = \{r_1, ..., r_H\}$
$\mathfrak{P}$	Set of the periods, $\mathfrak{C} = \{p_1, ..., p_M\}$
$\mathfrak{F}$	Set of the curricula, $\mathfrak{F} = \{f_1, ..., f_Q\}$
f_j	The j^{th} curriculum including a set of courses
b_i	The number of lectures of course c_i
b	The total number of all lectures $\sum_1^n b_i$
s_i	The number of students attending course c_i
rcp_k	The seat capacity of room h_k
t_i	The teacher taking course c_i
m_i	The number of minimum working days of course c_i
$Conf_{i,k}$	Whether course c_i and c_k are in conflict; 0, if $(t_k \neq t_i)$ $\wedge(\forall f_Q, c_k \notin f_Q \vee c_i \notin f_Q)$ 1, otherwise
$Unav_{i,k}$	if course c_i is available at period p_k. $unav_{i,k} = 0$ if it available, 1 otherwise.
X	The CB-CTT solution

In an attempt to choose a suitable timetable representation for IABC, all courses in CB-CTT solutions are mapped into particular periods and rooms. The representation is in the form of a matrix **X** with *n* rows and h columns, where *n* and h refers to the number of periods and the number of rooms, respectively. The value $x_{i,j}$ is the lecture that takes place in the period *i* and room *j*. The $x_{i,j}$ takes the value -1 once it is empty i.e. no lecture is placed in the position of the timetable corresponding to period *i* and room *j*. see Figure *1*.

$$\begin{array}{cc} \mathbf{X} = \begin{bmatrix} x_{11}, & x_{1,2} & x_{1,3} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & x_{23} & \cdots & x_{2,n} \\ x_{3,1} & x_{3,2} & x_{3,3} & \cdots & x_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{h,1} & x_{h,2} & x_{h,3} & \cdots & x_{n,h} \end{bmatrix} & \downarrow Rooms \\ \rightarrow Periods & \end{array}$$

Figure 1: Timetable Representation.

Basically, with this representation, the CB-CTT solution satisfies the Hard constraints H2 directly. The following is the mathematical formulations for the four hard and four soft constraints, is similar to the one used in [19]:

- **H1: Lectures** $\forall c_y \in \mathfrak{C}$

$$\sum_{i \in N} \sum_{k \in H} \textbf{\textit{valid}}(x_{i,k} = c_y) = b_y$$

where ***valid*** is a boolean function that return 1 if the b_y is in a position and 0 otherwise.

- **H2: Room Occupancy** The solution representation as shown in Figure 1. This hard constraint is satisfied automatically.
- **H3: Conflicts** $\forall x_{i,j}, x_{i,z} \in \mathbf{X}$,

$$x_{i,j} = c_a, x_{i,z} = c_b; \textit{Cflct}_{a,b} = 0$$

- **H4: Availability** $\forall x_{i,j} \in \mathbf{X} \wedge x_{i,j} = c_y$

$$uavb_{y,i} = 0$$

- **S1: Room Capacity:** $\forall c_y = x_{i,k}, x_{i,k} \in \mathbf{X}$.

$$f1(x_{i,k}) = \begin{cases} nsc_y - rcp_k, & rcp_k < nsc_y; \\ 0, & otherwise. \end{cases}$$

- **S2: Room Stability:** $\forall c_y \in \mathfrak{C}$,

$$f_2(c_y) = crs_y(\mathbf{X}) - 1$$

- **S3: Curriculum Compactness:** $\forall c_y = x_{i,k}, x_{i,k} \in \mathbf{X}$,

$$f_3(x_{i,j}) = \sum_{f_j \in \mathfrak{f}} valid c_y \in f_j \cdot \mathfrak{z}_{q,j}(\mathbf{X})$$

$$\mathfrak{z}_{q,i}(X) = \begin{cases} 1, & if \gamma \\ 0, & otherwise. \end{cases}$$

where $\gamma = if(i\%T = 1 \vee f_{q,i-1}(\mathbf{X}) = 0)$ $(i\%T = 0 \vee f_{q,i+1}(\mathbf{X}) = 0)$

- **S4: Minimum Working Days:** $\forall c_y \in \mathfrak{C}$,

$$f_4(c_y) = \begin{cases} cmd_y - nod_y(\mathfrak{X}), & if cmd_y < nod_y; \\ 0, & otherwise. \end{cases}$$

Based on the above formulation, The cost of violation of the four soft constraints can be calculated for given solution as shown in Equation 1. The weight for the violations of each soft constraints as given by the organisers of ITC-2007 are represented by $\partial_1 = 1, \partial_2 = 5, \partial_3 = 2$, and $\partial_4 = 1$.

$$f(\mathbf{X}) = \sum_{x_{i,k} \in \mathfrak{X}} \partial_1 \cdot f1(x_{i,k}) + \sum_{c_y \in \mathfrak{C}} \partial_2 \cdot f_2(c_y) + \sum_{c_y \in \mathfrak{C}} \partial_3 \cdot f_3(c_y) + \sum_{c_y \in \mathfrak{C}} \partial_4 \cdot f_4(c_y) \quad (1)$$

III. An improved Artificial Bee Colony Algorithm

This section describes the proposed Improved ABC algorithm. Firstly, a brief description of the ABC is provided, while secondly the process of optimising the proposed Improved ABC algorithm is presented.

A. Fundamentals of Artificial Bee Colony Algorithm

Karaboga [7] introduced a new ABC. This is a nature inspired metaheuristic algorithm which was motivated by imitating the intelligent foraging behaviour of honey bees. The performance was investigated using continuous optimisation problems. The idea is based on the model proposed in [23] for the exploration behaviours of honey bee colonies. The classifications of foraging artificial bees in ABC algorithm are grouped into three: employed bees areassociated with particular food sources, onlooker bees examine the dancing behaviour of employed bees in the hive, to choose the desired food source and scout bees search for food sources randomly once the employed get stuck with unsatisfactory food source. Both onlookers and scouts are referred to as unemployed bees. The new positions of all food sources are discovered by the scout bees initially. Thereafter, the exploitation of nectar of the food sources are carried out by employed bees and onlooker bees. The process of exploiting food sources continuously eventually cause them to become exhausted. Then, the employed bee which was exploiting the exhausted food source becomes a scout bee in search of other food source once again. In other words, the employed bee whose food source has been exhausted becomes a scout bee. In ABC, the position of a food source is the possible position of the solution to the problem and the nectar amount of a food source signifies the quality (fitness) of the associated solution. The number of employed bees is equal to the number of food sources (solutions) [7]. The algorithm has the ability for employing fewer control parameters and can compete well with other population-based algorithms [24], [25].

ABC algorithm have been applied to solve a variety of real world optimisation problems in the fields of engineering, AI and optimisation including structural and concrete analyses [26], protein folding simulation [27], quadratic knapsack problem [28], job shop problem [29], the lot-streaming flow shop scheduling problem [30], digital IIR filter [31] and reconfiguring distribution network [32].

The ABC algorithm starts by randomly generating a population of solutions using iterative process. The following 5 steps are repeated until termination condition is met.

- Send the scouts onto the initial food sources
- REPEAT
  - Send the employed bees onto the food sources and determine their nectar amounts
  - Calculate the probability value of the sources with which they are preferred by the onlooker bees
  - Send the onlooker bees onto the food sources and determine their nectar amounts
  - Abandon the exploitation process, if the sources are exhausted by the bees
  - Send the scouts into the search area for discovering new food sources, randomly
  - Memorise the best food source found so far
- UNTIL (requirements are met)

The detailed description of the steps can be found in [7],[25]

B. *An improved ABC for CB-CTT*

In an improved artificial bee colony algorithm (IABC), the neighbourhood structures are used by the operators such as employed and onlooker bees to explore different search space rigorously to enhance the quality of the solution. The idea of using more than one neighbourhood structures is to reduce the redundance or ineffectiveness of using a particular types alone. For example, the search space which can easily be reached by swap may be difficult for move to be accessed. The two neighbourhood structures used are move and swap, denoted by NL-move and NL-swap respectively:

1) $NL-Move$: moves selected course to a feasible period and room randomly.
2) $NL-Swap$: swap of two selected courses at random.

1) ***Population Generation:*** To ensure all population feasibility and diversity in the ABC algorithm, an initial population of timetable is generated with the use of saturation degree (SD) followed by backtracking algorithm (BA). The BA was used to alleviate the problems encountered by SD in our previous work when it failed to assign all the courses into periods and rooms in four problem instances [8]. In an IABC, SD starts with an empty timetable, where the course with the least number of valid periods in the schedule is assigned first as shown in [33]. The next selected course to be scheduled is based on the number of available periods; where it may not be able to scheduled all lectures of a course into the periods and rooms because of earlier assignments by SD, then the backtracking algorithm (BA) is applied to re-assign unscheduled courses. This technique guarantees that all the solutions are feasible.

Firstly, all courses that cannot be scheduled to the timetable would be entered into a list called A. The backtracking algorithm would then select each unassigned course (c) from the list and explore all courses in conflict in the timetabling. Those courses which are in conflict with course (c) are removed from the timetable and added to A again. Subsequently, the backtracking algorithm attempts to re-assign periods and rooms to all courses in the list. This process is repeated several times until no further courses could be scheduled, in which case the SD, followed by the BA processes are repeated from the beginning, until all the courses are scheduled. The pseudocode for generating timetable solutions is given in Figure 2.

Figure 2 The pseudocode for generating timetable solutions

```
while j ≤ N do
  repeat $x^j = \phi$
  Saturation Degree($x^j$)
  if($x^j$ is not complete and predefined iterations are not met)
    Backtracking ($x^j$)
  until ($x^j$ is complete)
  save $x^j$ in SN
  evaluate $f(x^j)$
endwhile
```

2) ***Employed bee phase:*** Here, the employed bees operator select a timetable solution from the population one by one and applies the two neighbourhood structures randomly to generate other solutions from the neighbourhoods. The fitness of the new solution is calculated if it is better than that of previous solutions, then the best solution with higher fitness is memorised. This process is repeated iteratively until all solutions have examined. This is done by using the two neighbourhood structures i.e. NL-move and NL-swap as shown in Figure 2.

Figure 3 The pseudocode for employed bee operator

```
for j = 1 ··· SN do
begin
    $\mathfrak{Y} = (y_1, y_2, \cdots, y_{SN})$
    rnd = U(0,1)
    if(rnd $\leq$ 0.7)
        NL-Swap
    else
        NL-move)
    endif
end
```

*3) **Onlooker bee phase**:* Onlooker bees select a timetable solution y_i using tournament selection technique. The tournament selection of size two is used because of its ability to escape local optima solution [34]. For example, in tournament selection, two solutions are selected randomly by the onlooker bee and their fitness is compared, then the fittest solution is chosen. Similarly, onlooker applies the neighbourhood structure in the same manner as used by the employed bee. The fitness of the new solution is calculated and if it is better than the current one, it replace it.

*4) **Scout bee phase**:* This is known to be the colony explorer. It works once a solution is abandoned, i.e. if a solution in the memory is not improved for certain number of iterations. ABC generates a new solution randomly and substitutes the abandoned one.

*5) **Memorise Best Food Source:**:* The fitness cost of the solutions is evaluated by the ABC algorithm. It memorize the best amongst them.

*6) **Stop Condition**:* Here, the ABC algorithm repeats the process until the maximum cycles number parameter is met.

IV. Computational Results and Discussions

The proposed improved ABC algorithm was programmed using Visual C++ 6.0 on an Intel machine with Core^{TM}, a 2.66GHz processor and a 2GB RAM. The efficiency of the algorithm was tested on 21 standard benchmark curriculum based course timetabling problem provided by ITC-2007 with 5 runs per problem instance, for the purpose of statistical calculation. The parameters used in the proposed algorithm are shown in Table 3. The overall fitness cost for each dataset was evaluated using the objective function formulation in equation (1), which sums up all the soft constraints violations. Note that feasibility is kept.

Table 3. ABC algorithm parameter setting

Parameter	MCN	SN	Limit
Value	10000	100	1000

Table 4. shows the results obtained by the proposed IABC in the first column, the original ABC in the second column and the percentage of improvements in the third column. From Table 4, it could be noted thatNotably, the IABC achieved results with better quality than basic ABC in most of the problem instances under the same parameter settings.

Table 4. Experimental Results of ABC

Problem	IABC	BASIC ABC	% Improvement
Comp01	**24**	-	-
Comp02	**299**	312	4.34
Comp03	**270**	292	10.74
Comp04	**166**	193	16.26
Comp05	**456**	-	-
Comp06	**255**	336	31.76
Comp07	**253**	324	28.06
Comp08	**173**	218	26.02
Comp09	**271**	302	11.44
Comp10	**239**	274	14.64
Comp11	**220**	293	33.18
Comp12	**751**	-	-
Comp13	**214**	-	-
Comp14	**221**	236	6.78
Comp15	**238**	284	19.32
Comp16	**236**	281	19.06
Comp17	**280**	331	18.21
Comp18	**173**	196	13.29
Comp19	**276**	304	10.15
Comp20	**241**	372	54.35
Comp21	**364**	-	-

The percentage of improvement shows that an improved ABC is able to enhance the solution quality for all problem instances, except in Comp02 and Comp14, where the percentage of improvement is less than 10% each.

Table 5 shows the average results of the top five competitors for CB-CTT in track 3 of the ITC 2007, as reported in [15]. The results are arranged in the order of the overall ranking, starting with the results of IABC in the leftmost column (T1) followed by the overall winner of the competition, Thomas M$\ddot{u}$ller (T2). The best results produced by the improved ABC as shown in Table 5 are close to previous cited results, although not but not comparatively better.

Table 5. The top five competitor's average result of CB-CTT, ITC-2007

Problem	T1	T2	T3	T4	T5	T6
Comp01	24	5.0	5.0	5.1	6.7	27.0
Comp02	299	61.3	61.2	65.6	142.7	131.1
Comp03	270	94.8	84.5	89.1	160.3	138.4
Comp04	166	42.8	46.9	39.2	82.0	90.2
Comp05	456	343.5	326.0	334.5	525.4	811.5
Comp06	255	56.8	69.4	74.1	110.8	148.9
Comp07	253	33.9	41.5	49.8	76.6	153.4
Comp08	173	46.5	52.6	46.0	81.7	96.5
Comp09	271	113.1	116.5	113.3	164.1	148.9
Comp10	239	21.3	34.8	36.9	81.3	101.3
Comp11	220	0.0	0.0	0.0	0.3	5.7
Comp12	751	351.6	360.1	361.6	485.1	445.3
Comp13	214	73.9	79.2	76.1	110.4	122.9
Comp14	221	61.8	65.9	62.3	99.0	105.9
Comp15	238	94.8	84.5	89.1	160.3	138.0
Comp16	236	41.2	49.1	50.2	92.6	107.3
Comp17	280	86.6	100.7	107.3	143.4	166.6
Comp18	173	91.7	80.7	73.3	129.4	126.8
Comp19	276	68.8	69.5	79.6	132.8	125.4
Comp20	241	34.3	60.9	65.0	97.5	179.3
Comp21	364	108.0	124.7	138.1	185.3	185.8

Where:

- **T1**: An improved ABC
- **T2**: A constraint-based solver by Thomas M$\ddot{u}$ller (2008)
- **T3**: Solving the course timetabling problem L$\ddot{u}$ and Hao (2008)

- **T4**: Incorporating Tabu search and Iterated local search by Atusta et.al (2008)
- **T5**: Application of the threshold accepting by Geiger (2008)
- **T6**: Quick-Fix repair based timetable by Clark et.al (2008)

The results produced by the IABC algorithm for CB-CTT problem is better than the original ABC using the same parameter settings. However, the achieved results are still not comparable with the best available in the literature. Further enhancement currently ongoing, to incorporate advanced neighbourhood structures such as neighbourhood union, token ring etc. These are necessary, to further improve the performance of the ABC algorithm for desired outcomes to be comparable or even better than those in the literature.

V. Conclusion and Possible Future Directions

This paper proposed an improved ABC algorithm for CB-CTT. The incorporation of 2 neighbourhood structures to the ABC enabled new solutions are generated from the neighbourhood. The improved ABC enhanced the results to be better than the original ABC, which was tested using ITC-2007, suggesting that the algorithm is capable of solving timetabling problems. Although, the improved ABC is able to yield good quality results, yet these are not comparable with the best results obtained by other methods. Further improvement are being exploited for enhancement.

Acknowledgment

This research was partially supported by the Universiti Sains Malaysia Postgraduate Fellowship Scheme 2011, awarded to the first author.

References

[1] M. Malim, A. Khader, and A. Mustafa, "Artificial immune algorithms for university timetabling," in *Proceedings of the 6th international conference on practice and theory of automated timetabling*. Citeseer, 2006, pp. 234–245.

[2] K. Socha, M. Sampels, and M. Manfrin, "Ant algorithms for the university course timetabling problem with regard to the state-of-the-art," *Applications of evolutionary computing*, pp. 334–345, 2003.

[3] X. Yang, "Firefly algorithms for multimodal optimization," *Stochastic Algorithms: Foundations and Applications*, pp. 169–178, 2009.

[4] S. Abdullah, H. Turabieh, B. McCollum, and E. Burke, "An Investigation of a Genetic Algorithm and Sequential Local Search Approach for Curriculum-based Course Timetabling Problems," 2010.

[5] M. Al-Betar and A. Khader, "A harmony search algorithm for university course timetabling," *Annals of Operations Research*, pp. 1–29, 2008.

[6] S. Irene, S. Deris, and M. Zaiton, "A study on PSO-based university course timetabling problem," in *International Conference on Advanced Computer Control*. IEEE, 2009, pp. 648–651.

[7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," *Techn. Rep. TR06, Erciyes Univ. Press, Erciyes*, 2005.

[8] L. B. Asaju, A. T. Khader, M. A. Al-betar, and A. Mohammed, "Artificial Bee Colony Algorithm for curriculum-based course timetabling problem," in *Fifth International Conference on Information Technology*. IEEE, 2011, pp. 546–552.

[9] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. IEEE, 1995, pp. 39–43.

[10] M. Garey and D. Johnson, *Computers and intractability. A guide to the theory of NP-completeness. A Series of Books in the Mathematical Sciences*. WH Freeman and Company, San Francisco, Calif, 1979.

[11] L. Di Gaspero and A. Schaerf, "Neighborhood portfolio approach for local search applied to timetabling problems," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 1, pp. 65–89, 2006.

[12] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. Parkes, L. Gaspero, R. Qu, and E. Burke, "Setting the research agenda in automated timetabling: The second international timetabling competition," *INFORMS Journal on Computing*, vol. 22, no. 1, pp. 120–130, 2010.

[13] G. Lach and M. Lübbecke, "Curriculum based course timetabling: Optimal solutions to the udine benchmark instances," *Burke and Gendreau (2008)*, 2008.

[14] E. Burke, J. Mareček, A. Parkes, and H. Rudová, "Ann Oper Res manuscript No.(will be inserted by the editor) A Branch-and-cut Procedure for the Udine Course Timetabling Problem," 2008.

[15] M. Geiger, "Applying the threshold accepting metaheuristic to curriculum based course timetabling," *Annals of Operations Research*, pp. 1–14.

[16] ——, "Multi-criteria Curriculum-Based Course Timetabling-A Comparison of a Weighted Sum and a Reference Point Based Approach," in *Evolutionary Multi-Criterion Optimization*. Springer, 2009, pp. 290–304.

[17] K. Shaker and S. Abdullah, "Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems," in *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*. IEEE, 2009, pp. 149–153.

[18] Z. Lü, J. Hao, and F. Glover, "Neighborhood analysis: a case study on curriculum-based course timetabling," *Journal of Heuristics*, pp. 1–22, 2009.

[19] Z. Lü and J. Hao, "Adaptive tabu search for course timetabling," *European Journal of Operational Research*, vol. 200, no. 1, pp. 235–244, 2010.

[20] ——, "Solving the Course Timetabling Problem with a Hybrid Heuristic Algorithm," *Artificial Intelligence: Methodology, Systems, and Applications*, pp. 262–273, 2008.

[21] T. Müller, "Itc2007 solver description: A hybrid approach," *Annals of Operations Research*, vol. 172, no. 1, pp. 429–446, 2009.

[22] L. Di Gaspero, B. McCollum, and A. Schaerf, "The second international timetabling competition (ITC-2007): Curriculum-based course timetabling (track 3)," in *Proc. of the 14th RCRA workshop on Exper. Eval. of Algo. for Sol. Prob. with Combinatorial Explosion, Rome, Italy*. Citeseer, 2007.

[23] D. Teodorović and M. DellOrco, "Bee colony optimization–a cooperative learning approach to complex transportation problems," in *Advanced OR and AI Methods in Transportation. Proceedings of the 10th Meeting of the EURO Working Group on Transportation, Poznan, Poland*. Citeseer, 2005, pp. 51–60.

[24] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[25] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[26] F. Kang, J. Li, and Q. Xu, "Structural inverse analysis by hybrid simplex artificial bee colony algorithms," *Computers & Structures*, vol. 87, no. 13-14, pp. 861–870, 2009.

[27] J. Chen, "Protein Folding Simulation Using a Modified Artificial Bee Colony Algorithm," 2009.

[28] S. Pulikanti and A. Singh, "An Artificial Bee Colony Algorithm for the Quadratic Knapsack Problem," in *Neural Information Processing*. Springer, 2009, pp. 196–205.

[29] B. Yao, C. Yang, J. Hu, G. Yin, and B. Yu, "An Improved Artificial Bee Colony Algorithm for Job Shop Problem," *Applied Mechanics and Materials*, vol. 26, pp. 657–660, 2010.

[30] Q. Pan, M. Fatih Tasgetiren, P. Suganthan, and T. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, 2010.

[31] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters," *Journal of the Franklin Institute*, vol. 346, no. 4, pp. 328–348, 2009.

[32] N. Linh and N. Anh, "Application Artificial Bee Colony Algorithm (ABC) for Reconfiguring Distribution Network," in *2010 Second International Conference on Computer Modeling and Simulation*. IEEE, 2010, pp. 102–106.

[33] D. Brélaz, "New methods to color the vertices of a graph," *Communications of the ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[34] L. Wang, "Shop scheduling with genetic algorithms," *Tsinghua University & Springer Press, Beijing*, 2003.