2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Novel Hybrid Swarm based Approach for Curriculum based Course Timetabling Problem

Cheng Weng Fong, Hishammudin Asmuni, Way Shen Lam, Barry McCollum, and Paul McMullan

Abstract—**This work applies a hybrid approach in solving the university curriculum-based course timetabling problem as presented as part of the 2nd International Timetabling Competition 2007 (ITC2007). The core of the hybrid approach is based on an artificial bee colony algorithm. Past methods have applied artificial bee colony algorithms to university timetabling problems with high degrees of success. Nevertheless, there exist inefficiencies in the associated search abilities in term of exploration and exploitation. To improve the search abilities, this work introduces a hybrid approach entitled nelder-mead great deluge artificial bee colony algorithm (NMGD-ABC) where it combined additional positive elements of particle swarm optimization and great deluge algorithm. In addition, nelder-mead local search is incorporated into the great deluge algorithm to further enhance the performance of the resulting method. The proposed method is tested on curriculum-based course timetabling as presented in the ITC2007. Experimental results reveal that the proposed method is capable of producing competitive results as compared with the other approaches described in literature.**

I. INTRODUCTION

TIMETABLING problems have been long classified as NP-hard combinatorial optimization problems and have attracted attention of researchers from the fields of Operational Research and Artificial Intelligence. This work focuses mainly on curriculum-based course timetabling problem, which deals with the assignment of lectures for courses offered by university into a specific number of timeslots so as to evade clashes and which are subject to variety of other side constraints. Both clashes and side constraints can be referred as hard and soft constraints, respectively. In general, hard constraints must be satisfied under any circumstance with violation of soft constraints acceptable but should be minimized. Timetable solutions that satisfy all the hard constraints are known as feasible solutions.

University course timetabling problems become complicated whenever there is an increase in the of number of courses offered by a university, an increase of number of students and indeed an increase of side constraints to be satisfied. In real world cases, it is usually impossible to satisfy all the soft constraints. Hence, efforts are made in maximizing the satisfaction on soft constraints in order to generate good quality solution. The earlier automated approaches for university timetabling were based on the graph coloring heuristics [1]. However, the performance of these approaches was usually poor and deteriorated as the complexity of the problem increase.

To enhance the efficiency of automation approaches, meta-heuristic approaches have been introduced and applied to university timetabling problems over the last two decades or so. Meta-heuristics can be divided into two types, i.e. single solution based and population based approaches. The former approaches manipulate (performing local changes on current solution until a local optima solution is obtained) a single solution during the search process. Hence, these approach usually posses' good exploitation ability which is beneficial in seeking for local optima solution. Examples of successful single solution based approaches that have been applied to university timetabling problems are simulated annealing [2, 3], great deluge algorithm [3, 4] and tabu search [5, 6]. Population based approaches manipulate a group of solutions during the search process which makes them effective in exploring different unknown search regions. Examples of such approaches are honey bee mating optimization algorithm [7], fish swarm optimization [4] and harmony search [8].

The Artificial Bee Colony algorithm (ABC) is a population based approach that mimics the food source searching behavior of honey bee colony. The exploitation ability is controlled by employed and onlooker bees by performing neighborhood search, while the exploration ability is controlled by the scout bee function by randomly seeking for a new solution in the search region. However, there are still some inefficiencies in term of the exploration and exploitation ability which degrades its performance [9]. Hence, a hybrid ABC algorithm is proposed with the aim for enhancing the search abilities of ABC algorithm.

The remaining sections of this work are organized as follow. Section II presents the information regarding the university curriculum-based course timetabling. Details on implementation of the basic ABC algorithm and the proposed method are covered in Section III. Experimental results are discussed in Section IV and lastly, conclusion and possible potential future works on this research work are presented in Section V.

II. PROBLEM DESCRIPTION

The curriculum-based course timetabling problem is one of the educational timetabling problems commonly studied by researchers in this area. It involves allocating lectures or other events associated with courses into a set of permitted timeslots and rooms on a weekly basic subject to satisfying a

Cheng Weng Fong, Hishammuddin Asmuni and Way Shen Lam are with Software Engineering Research Group, Software Engineering Department, Universiti Teknologi Malaysia, 81310 UTM Skudai, Johor, Malaysia. (email: chengweng0410@hotmail.com, hishamudin@utm.my lwshen007@hotmail.com)

Barry McCollum and Paul McMullan are with Department of Computer Science, Queen's University Belfast, Belfast BT7 1NN United Kingdom. (b.mccollum@qub.ac.uk, p.p.mcmullam@qub.ac.uk)

set of predefined constraints (hard and soft). The curriculum-based course timetabling problem investigated in this study was based on ITC2007 [10, 11] (third track) where there are curricula that manifest conflicts between any pair of courses in the curriculum.

The technical problem description for the curriculum-based course timetabling studied in this work is adopted from [6]. There are four hard constraints (H1 to H4) and four soft constraints (S1 to S4) considered in this problem and listed as below:

- H1 (Lectures) – All lectures of a course must be scheduled, and allocated into different periods. Violation is counted if a lecture remains unscheduled or two lectures within a course allocated in same period.
- H2 (Conflicts) – All lectures under same curriculum or taught by same teachers must be scheduled into distinct periods. Violation occurs if two conflicting lectures allocated in same period.
- H3 (Availability) – If a teacher not available in a particular period, then no any lectures can be allocated at that period. Violation is counted if a lecture scheduled in an unavailable period.
- H4 (Room Occupation) – Only one lecture can be allocated in a room at a period. Violation occurs if extra lectures scheduled in the same period and room.
- S1 (Room Capacity) – The room that assigned for a lecture must have enough seats to accommodate students that attend the course. Each student exceeds the capacity number counts as 1 violation.
- S2 (Minimum Working Days) – All lectures of a given course must spread over the minimum working days. Each day below the minimum days count as 1 violation.
- S3 (Isolated Lectures) – Lectures that under the same curriculum group should be adjacent to each other. Each isolated lecture count as 1 violation.
- S4 (Room Stability) – Lectures of a course should be take part in a same room. Each different room allocated to the lectures count as 1 violation.

The notation used in this problem is presented in Table I and the penalty calculations for hard and soft constraints violation are described as follow:

H1: Lectures: $\forall c_k \in C$,

$$\sum_{i=1,\dots,p=1,\dots,m} \chi\{x_{i,j} = c_k\} = l_k$$

Where χ is a truth indicator function which takes value of 1 if the given proposition is true, 0 otherwise.

H2: Conflict:

$$\forall x_{i,j}, x_{i,k} \in X, x_{i,j} = c_u, x_{i,k} = c_v$$

$$con_{uv} = 0$$

H3: Availability: $\forall_{i,j} = c_k \in X, uav_{k,i} = 0$

H4: Room occupancy: $\forall x_{k,j} \in X, x_{k,j} = c_i$

TABLE I

NOTATION USED FOR CURRICULUM BASED COURSE TIMETABLING DATASET

Symbol	**Description**
N	Total number of courses
M	Total number of rooms
D	Total number of working days per week
H	total number of timeslots per working day
P	Total number of periods, $p = d \times h$
S	Total number of curricula
C	Set of courses, $C = \{c1, \dots, cn\}$, $\lvert C\rvert = n$
R	Set of rooms, $R = \{r_1, \dots, c_m\}$, $\lvert R\rvert = m$
T	Set of periods, $T = \{t_1, \dots, t_p\}$, $\lvert T\rvert = p$
CR	Set of curricula, $CR = \{Cr_1, \dots, Cr_s\}$, $\lvert CR\rvert = s$
l_i	Number of lectures of course c_i
l	Total number of all lectures, $l = \sum_1^n l_i$
std_i	Number of students attending course c_i
tc_i	Teacher instructing course c_i
md_i	Number of minimum working days of course c_i
cap_i	Capacity of room r_j
$uav_{i,j}$	Whether course c_i is unavailable at period t_j. $uav_{i,j} = 1$ if it is unavailable, $uav_{i,j} = 0$ otherwise
con_{ij}	Wheter course c_i and c_j are conflict with each other; $con_{i,j} = \begin{cases} 0, & if\ (tc_i \neq tc_j) \wedge (\forall Cr_q, c \notin Cr_q \vee c_j \notin Cr_q) \\ 1, & otherwise \end{cases}$
x_{ij}	Course allocated at period t_i and room r_j
$nr_i(X)$	Number of rooms occupied by course c_i for a candidate solution X; $nr_i(X) = \sum_{j=1}^{m} \sigma_{ij}(X)$, where $\sigma_{ij} = \begin{cases} 1, & if\ \forall x_{kj} \in X, x_{kj} = c_{ij}, \\ 0, & otherwise \end{cases}$
$nd_i(X)$	Number of working days that course c_i take place at in candidate solution X; $nd_i = \sum_{j=1}^{d} \beta_{ij}(X)$, where $\beta_{ij} = \begin{cases} 1, & if\ \forall x_{u,v} \in X, x_{u,v} = c_i \wedge \lfloor u/h \rfloor = j \\ 0, & otherwise \end{cases}$
$app_{k,j}(X)$	Whether curriculum Cr_k appears at period t_i in candidate solution X; $app_{k,j}(X) = \begin{cases} 1, & if\ \forall x_{ij} \in X, x_{ij} = \mathrm{c_u} \wedge \mathrm{c_u} \in \mathrm{Cr_k}, \\ 0, & otherwise \end{cases}$

S1: Room capacity: $\forall x_{i,j} = c_k \in X$,

$$f_1(x_{i,j}) = \begin{cases} std_k - cap_j, if\ std_k > cap_j, \\ 0, otherwise \end{cases}$$

S2: Minimum working days: $\forall c_i \in C$,

$$f_2(C_i) = \begin{cases} md_i - nd_i,\ if\ nd_i(X) > md_i, \\ 0,\ otherwise \end{cases}$$

S3: Isolated lectures (curriculum compactness):

$$\forall x_{i,j} = c_k \in X,$$

$$f_3(x_{i,j}) = \sum_{Cr_q \in CR} \chi\{c_k \in Cr_q\} \cdot iso_{q,i}(X),$$

$$iso_{q,i} = \begin{cases} 1, if\ a \wedge b \\ 0, otherwise \end{cases}$$

Where,

$$a = i \bmod h = 1 \vee app_{q,i-1}(X) = 0,$$
$$b = i \bmod h = 0 \vee app_{q,i-1}(X) = 0$$

S4: Room stability: $\forall c_i \in C$,

$$f_4(c_i) = nr_i(X) - 1$$

The penalty cost calculation is summarized as (1):

$$f(X) = f1 + f2 + f3 + f4 \quad (1)$$

Where,

$$f1 = \sum_{x_{i,j} \in X} \alpha_1 \cdot f_1(x_{i,j}),$$
$$f2 = \sum_{c_i \in C} \alpha_2 f_2(c_i),$$
$$f3 = \sum_{c_i \in C} \alpha_3 \cdot f_3(c_i),$$
$$f4 = \sum_{x_{i,j} \in X} \alpha_4 \cdot f_4(x_{i,j})$$

For $\alpha_1, \alpha_2, \alpha_3$ and α_4 , each of them represents penalty weighted that associated with each soft constraint. In curriculum-based course timetabling, all these values are initialized as follow [11]: $\alpha_1 = 1, \alpha_2 = 1, \alpha_3 = 5$ and $\alpha_4 = 2$.

III. THE ALGORITHM

A. Artificial Bee Colony Algorithm

The main idea of ABC algorithm is to mimic the foraging process of the honey bee colony around the bee hive. It was firstly introduced by Karaboga [12] in addressing numerical optimization problems. Generally, three types of bee collaborate in the foraging process, i.e. employed bees, onlooker and scout bees. During the food searching process, employed bees will search for new food sources around the bee hive. The amount of nectar of food sources will be examined and information will be shared with the onlooker bees by performing a "waggle dance" once employed bees return to the bee hive. Onlooker bees will select the advertised food sources and search for new food sources around the selected food sources. If any of the food sources are exhausted, scout bees will search for new food sources without any information.

In ABC algorithm, the food source and nectar represent the potential solution and penalty cost value of the potential solution, respectively. During the search process, employed bees will explore for food sources in the search region and they will share the information of the explored food sources with the onlooker bees. Onlookers will select the food sources based on a Roulette Wheel Selection (RWS) scheme and further improve the quality of selected food sources. RWS is a probability selection scheme where food sources with better quality of nectar will have higher chances to be selected by onlooker bees. If a food source is exhausted, the employed bee will abandon the food source and turn into scout to explore new food source in the search region. It should be noted that the employed and onlooker bees search for food sources by using neighborhood search (examine neighborhood solution of current solution).

B. Great Deluge Algorithm

The Great Deluge (GD) algorithm is a local search approach that introduced by Dueck [13]. It works similar to simulated annealing to extent large degree where it accepts moves with positive feedback (improving solution) as well as negative feedback (worsening solution) if the quality of the solution is not worse than an adopted level. The value of level is initially assigned based on the penalty value of a solution and will decrease gradually across the search process. Prior to the execution of GD, an estimated quality (quality of desire final solution) need to be initialized by the user. GD will then explore towards search region which solutions having penalty values which are the same with estimated quality.

Note that there is no guarantee in obtaining final solution with same penalty value as the estimated quality. This means that the final solution might having penalty value that is equal, better or worse than the estimated quality. Furthermore, the decay rate of the level depends on the estimated quality. In this work, the value of estimated quality is determined by using the NMSS algorithm which discussed in Section III.C.

C. Nelder-Mead Simplex Search

Nelder-Mead Simplex Search (NMSS) is a traditional local search that was proposed by Nelder and Mead [14] in solving unconstraint optimization problem. The method is based on the theory of simplex which is a geometrical figure that consists of $N + 1$ vertices in N dimension. Different dimension constitutes different types of simplex. For example, a two dimension gives a triangle simplex while a three dimension constitutes a tetrahedron simplex. There are four operations used to improve the simplex, i.e. reflection, expansion, external contraction and shrinkage. In this work, a triangle simplex is selected and three operations are used as below:

$$c_{cent} = f(x_{best}) + f(x_{\mathrm{sec\_}worst}) + f(x_{worst})/3 \quad (2)$$
$$\text{External\_Contraction } (EC) = [c_{cent} - (\beta \times c_{cent})] \quad (3)$$
$$\text{Reflection } (R) = [EC - (\alpha \times c_{cent})] \quad (4)$$
$$\text{Expansion } (E) = [R - (\gamma \times c_{cent})] \quad (5)$$

There are three coefficients need to be initialized which are external contraction coefficient β, reflection coefficient α and expansion coefficient γ. All these coefficients are used to control the gaps between the EC, R and E. The c_{cent} is calculated using the best, second worst and worst solutions in the population [15]. The values of EC, R and E ($EC > R > E$) is then calculated using (2-5). Three more regions are formed in between $EC - R$ ($EC1$, $EC2$ and $EC3$) and $R - E$ ($R1$, $R2$ and $R3$). These values represent different search regions in the search space and will be served as estimated quality for GD.

```
Initialization
Calculate estimated quality based on NMSS (External Contraction EC,
Reflection R and Expansion E using (2-5));
Calculate estimated qualities of EC1, EC2, EC3 and R1, R2, R3 from
EC – R and R – E, respectively.
Set total number of iterations for GD, GD_Iteration;

Improvement
For i = 1 to SN do
    Set i as initial solution, sol_i;
    Set sol_i as current best solution, sol_best ← f(sol_i);
    Set initial level, level ← f(sol_i);
    Calculate decay rate (β_EC, β_EC1, β_EC2, β_EC3, β_R, β_R1, β_R2, β_R3, β_E,)
    based on estimated qualities, (f(sol_i) – estimated quality)/
    Iteration;
    For j = 1 to GD_Iteration do
        Set estimated quality for sol_i;
        Set decay rate based on estimated quality, β;
        Apply one of the neighborhood structures (Nb1 - Nb6) on sol_i
         to produce new solution sol_i*;
        Calculate penalty value for f(sol_i*);
        If f(sol_i*) < f(sol_best)
            sol_i ← sol_i*;
            sol_best ← sol_i*;
        Else
            If f(sol_i*) < level
                sol_i ← sol_i*;
        End If
        Level ← level – β;
    End For j

    If f(sol_i) < f(sol_golBest)
        sol_golBest ← sol_i;
    End If

    If no improvement on quality of sol_i
        Increase BeelimitCount by 1;
    End If
End For i
```

Fig. 1. The pseudo code for the NMGD

D. Nelder-Mead Great Deluge Algorithm

In this work, the combination of NMSS with GD (NMGD in short) is incorporated into the basic ABC algorithm with the aim of enhancing search ability of ABC. The pseudo code for NMGD is presented in Fig. 1. Prior to the execution of GD, the NMSS is used to calculate estimated qualities for GD by using (2-5). After that, GD will intelligently select an appropriate estimated quality and decay rate for current solution and the solution improvement process begins. If the quality of the solution is better or equal to the selected estimated quality and the termination criteria is not met, another estimated quality will be selected and the solution searching process continues. This solution searching process is repeated for all the solution in the population and a new set of estimated qualities and decay rates will be recalculated in the next cycle of the solution searching process. By using various estimated qualities in GD, the algorithm is able to better fine tuning different search regions in reaching local optima solution. Six neighborhood moves that introduced by Muller [16] are employed in NMGD in generating neighborhood solution, i.e.

Nb1: Change the period of a randomly selected lecture.

```
Initialization
Set population size, SN;
Initialized the population and calculate penalty value for each
solution;
Identify best solution in the population, sol_golBest;
Set the iteration number for NMGD-ABC, NMGD_ABCNumIter;
Set the value for parameter limit, limit;
Set bee limit counter for each solution, BeeLimitCounter ← 0;

Improvement
For i = 1 to NMGD_ABCNumIter do

    For j = 1 to SN do    %Employed bee phase%
        If (sol_j is new solution generated by scout)
            Incorporate information of best solution into sol_j;
        End If
    End For j

    Improve solutions using NMGD;   %Onlooker bee phase%

    For j = 1 to SN do   %Scout bee phase%
        If (BeeLimitCounter of  sol_j > limit)
            Generate new solution (sol_new) and replace the old solution
            (sol_j), sol_j ← sol_new;
            Set BeeLimitCounter of  sol_j to 0, BeeLimitCounter ← 0;
        End If
    End For j
End For i
```

Fig. 2. Framework for NMGD-ABC

Nb2: Change the room of a randomly selected lecture.
Nb3: Select a lecture and assigns a new period and room for selected lecture.
Nb4: Select a course and try to assign all lectures of selected course into a new selected room. This neighborhood tries to reduce violation of room stability soft constraint.
Nb5: Select a course that violates minimum working days penalty and move a lecture from a day (day that having two or more lectures are taught) to a day that the course is not scheduled. This neighborhood tries to minimize the violation of minimum working days soft constraint.
Nb6: Select a lecture and move to a random selected period and room.

During the execution, tentative solution is generated using one of the neighborhoods that selected randomly.

E. Nelder-Mead Great Deluge Artificial Bee Colony Algorithm

It should be mentioned that both exploration and exploitation are important search abilities for evolutionary algorithm. With exploration ability, the search manages to examine various unknown regions and for exploitation ability, the search able to search for local optima solution for a particular search region. In this section, Nelder-Mead Great Deluge Artificial Bee Colony (NMGD-ABC) is proposed with the aim of improving the exploration and exploitation abilities of the solution searching process.

The proposed NMGD-ABC is divided into initialization and improvement phases as demonstrated at Fig. 2. At the former phase, a group of feasible solutions are constructed using constructive heuristic as in [17] and the penalty value of each solution is also calculated. In the latter phase, there are three sub-phases which cover employed bee, onlooker bee and scout bee phases.

In employed bee phase, the neighborhood search used in employed bee of basic ABC has been replaced by the global best concept inspired from Particle Swarm Optimization (PSO) [18]. Recall that the global best concept in PSO directs the search process in exploring around the promising regions. Hence, the implementation of global best concept in employed bee phase is with the aim of enhancing the search efficiency around the promising regions. The detail implementation of global best concept in ABC algorithm can be seen at author's previous work [9].

With reference to Fig. 2, NMGD is implemented in onlooker bee phase. In this phase, onlooker bees will exploit all the food sources using NMGD which described in Section III.D. It should be noted that the RWS scheme is eliminated in the proposed approach. This is with the hope of eliminating the relying on probability selection scheme that giving more attention on better quality of solutions which might mean an imbalance within exploitation.

Food sources that are exhausted will be abandoned and employed bees will turn into scout bees to explore for new food sources.

TABLE IV
RESULTS COMPARISON BETWEEN BASIC ABC, PROPOSED APPROACH AND BEST KNOWN IN THE LITERATURE

Dataset	Basic ABC		Proposed Approach		Best known
	600 (sec)	5000 iteration	600 (sec)	5000 iteration	
comp01	121	43	**5**	**5**	**5**
comp02	332	191	87	51	**29**
comp03	375	232	126	90	**66**
comp04	289	131	81	39	**35**
comp05	1352	941	776	361	**292**
comp06	394	267	182	64	**68**
comp07	321	292	68	21	**6**
comp08	342	162	132	53	**38**
comp09	461	211	191	121	**96**
comp10	302	203	152	36	**40**
comp11	219	64	21	**0**	**0**
comp12	711	535	462	377	**310**
comp13	332	191	141	79	**59**
comp14	293	169	163	61	**51**
comp15	377	265	171	91	**68**
comp16	346	236	152	59	**22**
comp17	354	229	142	101	**60**
comp18	276	159	103	90	**65**
comp19	343	211	173	81	**57**
comp20	443	231	163	42	**4**
comp21	459	257	212	124	**86**

IV. SIMULATION RESULTS

Experiments in testing performance of proposed approach has been conducted using curriculum-based course timetabling problem that are described in Section II. The proposed approach was coded using C++ programming language and simulations were carried out on 2.2 GHz laptop. The parameters for the proposed algorithm are shown in Table II. All the parameters setting are selected based on some preliminary experiments. It should be noted that this set of parameters setting should not be regarded as optimal setting, but a generalized one which allow the proposed method performs fairly well across all instances with different complexity. An example that demonstrates the estimated quality selection for NMGD in solving instance comp10 is presented in Table III. At 1st iteration, the penalty value is 2032, thus the estimated quality of *EC* will be selected. When comes to 1000th iteration, the penalty value has been reduced to 1885, therefore, estimated quality of *EC3* will be selected.

TABLE II
PARAMETERS SETTING FOR NMGD-ABC

No.	Parameters	Values
1	Termination criteria for basic ABC	600 sec / 5000 iterations
2	Termination criteria for NMGD-ABC	600 sec / 5000 iterations
3	Iteration number for NMGD-ABC	2000
4	*limit* (scout bee) for basic ABC	100
5	*limit* (scout bee) for NMGD-ABC	4
5	External Contraction coefficient	0.05
6	Reflection coefficient	0.05
7	Expansion coefficient	0.05
8	Crossover point	8

As referred to Table IV, it can be seen that two sets of experiments with different termination criteria were carried out, i.e. termination with 600 seconds computational time (30 runs for each instance), which is the competition rule in ITC2007 and 5000 iterations number (10 runs for each instance). The latter was an extended experiment with the aim of demonstrating proposed approach capable of producing better quality of solutions given extra processing time.

Table IV demonstrates the results comparison between basic ABC, proposed approach and the best known results in the literature (best solutions are highlighted in bold font). In general, the proposed approach outperforms basic ABC and capable of producing competitive results as compared with

TABLE III
ESTIMATED QUALITY SELECTION FOR NMGD

GD_Iteration	Penalty Value	Estimated Quality								
		EC	*EC1*	*EC2*	*EC3*	*R*	*R1*	*R2*	*R3*	*E*
		1956.67	1932.21	1907.75	1883.29	1858.83	1835.60	1812.36	1789.13	1765.89
sol_1										
1	2032	√								
2	1997	√								
…										
1000	1885				√					
…										
2000	1783									√

TABLE V
RESULTS COMPARISON BETWEEN BASIC PROPOSED APPROACH AND APPROACHES PUBLISHED IN THE LITERATURE

Dataset	NMGD-ABC	P1	P2	P3	P4	P5
comp01	**5**	**5**	**5**	**5**	9	-
comp02	51	**34**	50	43	103	312
comp03	90	**70**	82	72	101	292
comp04	39	38	**35**	**35**	55	193
comp05	361	**298**	312	**298**	370	-
comp06	64	47	69	**41**	112	336
comp07	21	19	42	**14**	97	324
comp08	53	43	40	**39**	72	218
comp09	121	**99**	110	103	132	302
comp10	36	16	27	**9**	74	274
comp11	**0**	**0**	**0**	**0**	1	293
comp12	377	**320**	351	331	393	-
comp13	79	**65**	68	66	97	-
comp14	61	**52**	59	53	87	236
comp15	91	**69**	82	84	119	284
comp16	59	38	40	**34**	84	281
comp17	101	**80**	102	83	152	331
comp18	90	**67**	68	83	110	196
comp19	81	**59**	75	62	111	304
comp20	42	35	61	**27**	144	372
comp21	124	105	123	**103**	169	-

TABLE VI
STATISTICAL ANALYSIS FOR BASIC ABC AND INMGD-ABC ON ITC2007 DATASET

Dataset	Basic ABC	NMGD-ABC	t-test p-value
comp01	43	5	< 0.05
comp02	191	51	< 0.05
comp03	232	90	< 0.05
comp04	131	39	< 0.05
comp05	941	361	< 0.05
comp06	267	64	< 0.05
comp07	292	21	< 0.05
comp08	162	53	< 0.05
comp09	211	121	< 0.05
comp10	203	36	< 0.05
comp11	64	0	< 0.05
comp12	535	377	< 0.05
comp13	191	79	< 0.05
comp14	169	61	< 0.05
comp15	265	91	< 0.05
comp16	236	59	< 0.05
comp17	229	101	< 0.05
comp18	159	90	< 0.05
comp19	211	81	< 0.05
comp20	231	42	< 0.05
comp21	257	124	< 0.05

the best known results. Indeed, the proposed approach with 600 seconds computational time managed to produce solution with penalty which equal with best known for instance comp01. For the extended experiment, the proposed approach manages to produce optimal solutions for instances comp01 and comp11. In addition, proposed approach also performs well on remaining instances. However, the price to pay for generating good quality solutions is larger amount of computational time required.

Results comparison with the approaches published in the literature is also carried out. The comparison is shown in Table V (best solutions are highlighted in bold font) and the selected methods in comparison are as below:

P1: Lu and Hao [6] – tabu search
P2: Geiger [19] – threshold acceptance method
P3: Muller [16] – constraint-based solver
P4: Clark et al. [20] – repair-based timetable solver
P5: Bolaji et al [21] – basic artificial bee colony algorithm

According to Table V, it can be seen that proposed approach capable of producing competitive results. Besides that, proposed approach is able to generate two optimal solutions on comp01 and comp11 instances as in P1, P2 and P3. As compared with P5, the results produced by proposed approach outperform P5 in all instances. This is due to the search abilities (in term of exploration and exploitation) of basic ABC are unable to search for feasible and better quality results ("-" indicates no feasible solution generated).

Fig. 3 presents the box and whisker plot of proposed approach with 5000 iterations number for instances comp03, comp06, comp14, comp 18 and comp21. It can be observed that the gaps between the average and best are smaller than the gaps between average and the worst. This implies that the proposed approach is robust and stable in solving different instances.

In order to demonstrate the effective of performance of the

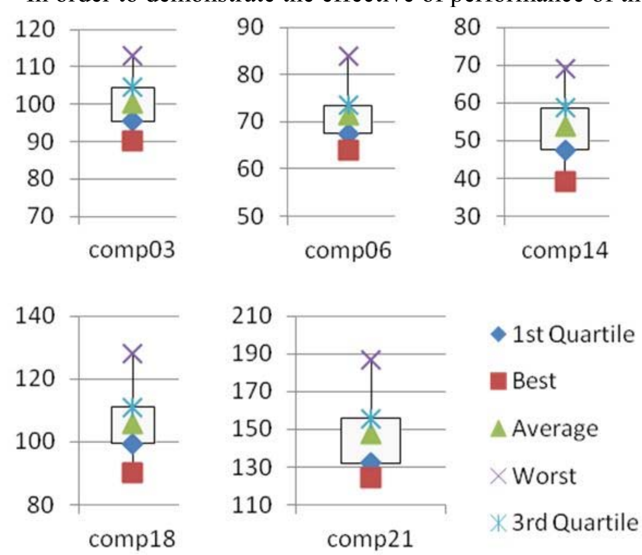

Fig. 3. Box and whisker plots of proposed approach for instances comp03, comp06, comp14, comp18 and comp21.

proposed approach over basic ABC algorithm, a statistical analysis is carried out by using the t-test as shown in Table VI. The null hypothesis is defined as there is no different in term of solution quality between basic ABC and proposed approach, while the alternative hypothesis is defined as there is different between both approaches. The level of confidence is 0.05 (95%).

By referring to the p-values in Table VI, it is clear that the p-values for all instances are smaller than 0.05, in other words, there performance of proposed approach is significant different from basic ABC algorithm.

V. CONCLUSION

In this work, a hybrid ABC algorithm named NMGD-ABC has been presented and tested on the curriculum based course timetabling introduced as presented in ITC2007. A hybridization of NMSS with GD (NMGD) is incorporated into ABC algorithm to replace the neighborhood search within the onlooker bee phase with the aim of improving the exploitation ability of ABC algorithm. In addition, an advantage of NMGD over GD is that NMGD manages to exploit different qualities of solutions with different values of estimated quality.

Inspired by PSO, global best concept is implemented into the employed bee phase. With this concept, the search process will explore toward promising search region and this increases the chances in reaching better quality solutions.

In short, the proposed approach is capable of generating comparative results as compared with the approaches published in the literature. The performance of proposed approach can be enhanced by introducing other exploration mechanism and this is subject to future work.

References

[1] R. Qu, and E. K. Burke, "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems," *Journal of the Operational Research Society,* vol. 60, no. 9, pp. 1273-1285, 2008.

[2] J. Thompson, and K. Dowsland, "Variants of simulated annealing for the examination timetabling problem," *Annals of Operations Research,* vol. 63, no. 1, pp. 105-128, 1996/02/01, 1996.

[3] E. K. Burke, Y. Bykov, J. Newall, and S. Petrovic, "A time-predefined local search approach to exam timetabling problems," *Yugoslav Journal of Operations Research,* vol. 13, pp. 139-151, Oct, 2003.

[4] H. Turabieh, and S. Abdullah, "An integrated hybrid approach to the examination timetabling problem," *Omega - The International Journal of Management Science,* vol. 39, no. 6, pp. 598-607, 2011.

[5] S. Abdullah, and H. Turabieh, "On the use of multi neighbourhood structures within a tabu-based memetic approach to university timetabling problems," *Information Sciences,* vol. 191, no. 0, pp. 146-168, 2012.

[6] Z. Lü, and J.-K. Hao, "Adaptive Tabu Search for course timetabling," *European Journal of Operational Research,* vol. 200, no. 1, pp. 235-244, 2010.

[7] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "A honey-bee mating optimization algorithm for educational timetabling problems," *European Journal of Operational Research,* vol. 216, no. 3, pp. 533-543, 2012.

[8] M. A. Al-Betar, A. T. Khader, and M. Zaman, "University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 42, no. 5, pp. 664-681, 2012.

[9] F. C. Weng, and H. Asmuni, "An Automated Approach Based On Bee Swarm in Tackling University Examination Timetabling Problem," *International Journal of Electrical and Computer Sciences,* vol. 13, no. 02, pp. 8-23, 2013.

[10] B. McCollum, P. McMullan, A. J. Parkes, E. K. Burke, and R. Qu, "A new model for automated examination timetabling," *Annals of Operations Research,* vol. 194, no. 1, pp. 291-315, 2012.

[11] B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu, and E. K. 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.

[12] D. Karaboga, *An idea based on honey bee swarm for numerical optimization,* Technical Report TR06, Erciyes University, 2005.

[13] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *Journal of Computational Physics,* vol. 104, no. 1, pp. 86-92, 1993.

[14] J. A. Nelder, and R. Mead, "A simplex method for function minimization," *The Computer Journal,* vol. 7, no. 4, pp. 308-313, 1965.

[15] 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.

[16] T. Müller, "ITC2007 solver description: a hybrid approach," *Annals of Operations Research,* vol. 172, no. 1, pp. 429-446, 2009/11/01, 2009.

[17] D. Landa-Silva, and J. Obit, "Evolutionary non-linear great deluge for university course timetabling," *Hybrid Artificial Intelligence Systems, 4th International Conference, Proceedings HAIS 2009, Lecture Notes in Computer Science*, E. Corchado, X. Wu, E. Oja, Á. Herrero and B. Baruque, eds., pp. 269-276, Salamanca, Spain: Springer, 2009.

[18] J. Kennedy, and R. Eberhart, "Particle swarm optimization." pp. 1942-1948 vol.4.

[19] M. J. Geiger, "An application of the threshold accepting metaheuristic for curriculum based course timetabling."

[20] M. Clark, M. Henz, and B. Love, "Quikfix a repair-based timetable solver."

[21] A. R. Yildiz, "A new hybrid artificial bee colony algorithm for robust optimal design and manufacturing," *Applied Soft Computing,* vol. 13, no. 5, pp. 2906-2912, 2013.