Artificial Bee Colony Algorithm for Curriculum-Based Course Timetabling Problem

Asaju La'aro Bolaji[a] , Ahamad Tajudin Khader[a] ,Mohammed Azmi Al-betar[a,b] , Mohammed Awadallah[a]
[a] School of Computer Sciences, Universiti Sains Malaysia, 11800, Pulau Pinang
Email: $abl10 - sa0739@student.usm.my$
[b] Faculty of Computer Science, Al-Zaytoonah Private University of Jordan, Amman 11733, Jordan

***Abstract*—This research article presents the adaption of the Artificial Bee Colony algorithm for solving timetabling problems, with particular focus on the curriculum-based course timetabling that formed part of the competition track 3 of the 2nd International Timetabling Competition in 2007 (ITC-2007). An attempt to solve these problems was made via an approach broken down into two parts; first, Saturation Degree (SD) was used to ensure a feasible solution, where the hard constraints are satisfied. Secondly, Artificial Bee Colony Algorithm was used to further improve the results obtained. The algorithm produced very good results, though they were not comparatively better than those previously reported in the literature due to the fact that the algorithm easily gets stuck in the local optimal solution. With proper modification and hybridizing local search-based algorithms this approach could make the algorithm perform better on timetabling problem in general.**

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

I. Introduction

Timetabling is an NP-hard combinatorial optimisation problem [1] which has captured the interest of many researchers in operational research and the artificial intelligence domain. University course timetabling problem (UCTP), the principal interest of this paper, is the assignment of a set of events to given periods and rooms subject to some given (hard and soft) constraints. Hard constraints must be satisfied in order to produce a feasible timetabling solution, yet satisfying the soft constraints is not mandatory but rather desired. The objective is to produce a feasible timetable with the least number of soft constraint violations.

Conventionally, UCTP normally has two forms: Post enrolment course timetabling and Curriculum-based course timetabling problem (CB-CTT). The CB-CTT has recently been proposed as part of the tracks of international timetabling competition (ITC-2007) [http://www.cs.qub.ac.uk/itc2007]. The aim of the competition was to bridge the gap between research and practice of the university timetabling domain [2]. Since then, several optimization algorithms have been developed to tackle the CB-CTT, but research in this area is still ongoing, due to the peculiar nature of the problem. Some of the developed algorithms include heuristic such as integer programming [3], [4], metaheuristic such as local search based and hybrid algorithms [5], [6], [7], [8], [9], [10].

The general description of CB-CTT and its formulations were presented in [11], where the insight and detailed overview of CB-CTT with respect to ITC-2007 were discussed. The integer programming approach also was proposed in [3], where the method used was broken down into two stages: the first stage was used to ensure the feasibility whereas in the second, the aim was to enhance the feasible solutions to the state-of-the-art results. The authors claimed that the algorithm was very robust in the sense that it steadily gave satisfactory lower and upper bound solution within a reasonable computation time without particular parameter tuning. Yet it did not emerge as the overall winner. Burke et.al in [4] used branch and cut technique to the CB-CTT problem; the process towards the solution in their work was divided into two stages. They presented an alternative integer programming formulation in the first phase, which allowed less than normal number of variables and mildly increased the number of soft constraints. The Branch and Cut procedure was presented in the second phase, where constraints from enumeration of events/free-period patterns, necessary to reach optimality, were added when they were violated. The results produced by the method showed that within 15 minutes, it was possible to find provably optimal solutions to two instances. The CB-CTT was studied in [12] that came up with a hybrid solution algorithm called Adaptive Tabu Search (ATS). Incorporated in ATS method was a new neighborhood and the combination of Tabu search with perturbation from iterated local search. The results showed that it improved the previously best known results. The ITC-2007 problem was considered by Müller [13] and was judged as the best in the CB-CTT. The method was a combination of great deluge and simulated annealing incorporated into the constraint solver library. The iterative local search was used in the construction stage and the solution was further enhanced with the aid of constraint-based statistics (CBS). Another method which used the local search approach for CB-CTT was presented in [5]. Here, a constructive heuristics which was similar to Squeaky Wheel optimization was used to obtain a feasible timetabling solution and Threshold Accepting rule from simulated annealing was later used to improve the solution. The computational result showed that the algorithm was good for the problem. In another study, the CB-CTT was treated as multi-objective optimization problem in [6]. The author presented a solution framework based on local search heuristics by using two different aggregation techniques i.e. a weighted sum aggregation and a reference point-based method. The experimental results produced showed that the approach was able to obtain good solutions.

The swarm intelligence is a branch of nature inspired algorithm which is aimed at simulating the behaviors of social insects and their activities in solving daily life problems within their environment. These kinds of algorithms are developed based on mathematical models and optimization techniques to solve real world optimization problems in a manner similar to that of social insects [14]. Examples of nature inspired algorithms as stated in [15] include Ant Colony Optimization Algorithm, Firefly Algorithm, Bat Algorithm, Fish schooling, Bee Algorithm, Bird Flock, Monkey Algorithm and Artificial Bee Colony algorithm, which is the main focus of this paper.

The main objective of this research is to adapt Artificial Bee Colony Algorithm (ABC) to CB-CTT as an initial work which explores the efficiency of the algorithm to solve the problem. The initial results show that ABC can efficiently produce good solutions but not comparable to those reported in the literature.

This paper is organized as follows; section two provides the detailed description and mathematical formulations of CB-CTT in terms of hard and soft constraints; details of Artificial Bee colony Algorithm are presented in section three; the ways in which ABC was adapted for CB-CTT are discussed in section four. Section five shows the computational results and their comparison with previous findings in the literature. The final section presents the conclusion and possible directions for the future.

II. THE CURRICULUM-BASED COURSE TIMETABLING

A. Problem Description

The CB-CTT problems deals with the assignment of a set of lectures of courses to a set of rooms and timeslots on a weekly basis, in accordance with a given set of constraints [11]. A timetable is considered feasible once all lectures of courses have been assigned to timeslots and rooms with respect to the (H1 - H4) hard constraints. In addition, a feasible timetable satisfying the four hard constraints gives a penalty cost for the violations of the four (S1 - S4) soft constraints, the main objective of the CB-CTT problem is to minimize the number of soft constraint violations in a feasible solution. The four hard constraints and four soft constraints are outlined below:

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 and 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 problem deal with an assignment of *tn* courses C=$(c_1, c_2........, c_{tn})$ to a set of *tr* rooms R=$(r_1, r_2........, r_{tr})$ and a set of *tp* periods P=$(p_1, p_2........, p_{tp})$, where period is the composition of minimum working days *td* and timeslots per working days *tt* i.e. $tp = td \times tt$. Each course c_i consists of a set of lectures l_i that has to be assigned to different periods. The problem consists of a set of curricula CU=$(Cu_1, Cu_2..., Cu_{tc})$ where each curriculum Cu_i is a group of courses having common students and finally, the CB-CTT consists of a set students std_i, where $std_i = (std_1, std_{2,} ...., Std_i)$ 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
tn	The total number of courses
tr	The total number of rooms
td	The total number of min. Working days per week
tt	The total number of timeslots per days
tp	The total number of periods, $tp = td \times tt$
tc	The total number of curricula
C	Set of the courses, $C = \{c_1, ..., c_{tn}\}$
R	Set of the rooms, $R = \{r_1, ..., r_{tr}\}$
P	Set of the periods, $P = \{p_1, ..., p_{tp}\}$
CU	Set of the curricula, $CU = \{Cu_1, ..., Cu_{tc}\}$
Cu_m	The m^{th} curriculum including a set of courses
l_i	The total number of lectures c_i
tl	The total number of all lectures $\sum_1^{tn} l_i$
std_i	The number of students attending course c_i
tch_i	The teacher taking course c_i
mwd_i	The number of minimum working days of course c_i
$rcap_k$	The capacity of room r_k
$unav_{i,j}$	Whether course c_i is available at period p_j. $unav_{ij} = 0$ if it available, $unav_{i,j} = 1$ otherwise.
$Conf_{i,j}$	Whether course c_i and c_j are in conflict; 0, if $(tch_i \neq tch_j)$ $\wedge(\forall Cu_q, c_i \notin Cu_q \vee c_j \notin Cu_q)$ 1, otherwise

From Table 1, it is possible to formulate the objective function in terms of hard and soft constraints. It is worthy to note that the objective function $f(X) = \sum(S_1+S_2+S_3+S_4)$, represents the total number of penalty cost for the violations of the four soft constraints. By this formulation, we assumed that hard constraints had been satisfied. The timetable solution $X_{tp,tr}$ for CB-CTT is represented by a matrix, where each element $x_{i,j}$ has either lecture *m* of course *c* or -1 if it is empty; i represents the timeslots or periods and j represents the room identity or number (as shown in Figure 1).

$$\begin{pmatrix} x_{0,0} & x_{0,1} & .... & x_{0,tr} \\ x_{1,0} & x_{1,1} & .... & x_{1,tr} \\ x_{2,0} & x_{2,1} & .... & x_{2,tr} \\ ...... & ...... & .... & .... \\ x_{tp,0} & x_{tp,1} & .... & x_{tp,tr} \end{pmatrix}$$

Figure 1, The CB-CTT Solution Representation

- **H1: Lectures** $\forall c_m \in C \sum_{i=1...tp,j=1...tr} sln_m(x_{i,j} = l_m)$
- **H2: Room Occupancy** The solution representation satisfied this hard constraint automatically.
- **H3: Conflicts** $\forall x_{i,j}, x_{i,l} \in X,\ x_{i,j} = c_a, x_{i,l} = c_b$ $Conf_{a,b}$ =0
- **H4: Availability** $\forall x_{i,j} \in X, x_{i,j} = c_m,$ $unav_{m,i}$ =0
- **S1: Room Capacity** $\forall x_{i,j} \in X, x_{i,j} = c_m$

$$OC_1(x_{i,j}) = \begin{cases} \beta_1 \cdot (std_m - cap_j), & if cap_j < std_m; \\ 0, & otherwise. \end{cases} \tag{1}$$

- **S2: Room Stability.** $\forall c_i \in C,$ $OC_2(c_i) = \beta_2 \cdot (tnr_i(X) - 1)$
- **S3: Curriculum Compactness.** $\forall x_{i,j} \in X, x_{i,j} = c_m,$

$$OC_3(x_{i,j}) = \beta_3 \cdot \sum_{cu_q \in CU} c - cu_{m,q} \cdot iso_{q,j}(X) \tag{2}$$

$$c - cu_{m,q} = \begin{cases} 1, & if c_m \in cu_q; \\ 0, & otherwise. \end{cases} \tag{3}$$

$$iso_{q,i}(X) = \begin{cases} 1, & if \gamma \\ 0, & otherwise. \end{cases} \tag{4}$$

where $\gamma = (i\%tt = 1 \vee cuw_{q,i-1}(X) = 0)$ $\wedge(i\%tt = 0 \vee cuw_{q,i-1}(X) = 0)$

- **S4: Minimum Working Days.** $\forall c_j \in C,$

$$OC_4(c_j) = \begin{cases} \beta_4 \cdot (d_1 - d_2), & if d_1 < d_2; \\ 0, & otherwise. \end{cases} \tag{5}$$

where $d_1 = mwd_j$ and $d_2 = nd_j(X)$

This formulation is adapted from [10] with minor modifications

The penalty value for the soft constraints are represented by $\beta_1, \beta_2, \beta_3, \beta_4$. The penalty weights for each were fixed for the each violations of soft constraints such as $\beta_1 = 1, \beta_2 = 1, \beta_3 = 5, \beta_4 = 2$. The objective cost $f(x)$ is represented as the summation of penalty values for the violation of four soft constraints, given as

$$f(X) = \sum_{x_{i,j} \in X} OCrc(x_{i,j}) + \sum_{c_i \in C} OC_{rs}(c_i) + \tag{6}$$

$\sum_{xi,j \in X} OC_{cc}(x_{i,i}) + \sum_{c_i \in C} OC_{mwd}(c_i)$

III. Artificial Bee Colony Algorithm

The Artificial Bee Colony (ABC) algorithm is a branch of nature inspired or swarm intelligence based meta-heuristic algorithm which was proposed by Karaboga [16] for optimizing numerical problems. It was motivated by the intelligent foraging behavior of honey bees. The algorithm is particularly based on the model proposed in [17] for the foraging manners of honey bee colonies. The model comprises three vital fundamentals: employed and unemployed foraging bees, and food sources. The first two fundamentals, employed and unemployed foraging bees search for rich food sources, which is the third fundamental, is being close to their hive. The two principal modes of behaviour are also described by the model, which are necessary for self-organization and collective intelligence: recruitment of foragers to the rich food sources resulting in positive feedback and abandonment of poor food sources by foragers causing negative feedback [16].

In ABC, the colony consists of three groups of bees: employed bees linked with specific food sources, onlooker bees studying the dancing behaviour of employed bees in the hive to choose the desired food source and scout bees searching for food sources randomly once the employed is stuck with unsatisfactory food source. Both onlookers and scouts are also known as unemployed bees. The positions of all food sources are discovered by the scout bees originally. Thereafter, the exploitation of nectar of food sources are carried out by employed bees and onlooker bees. The repetitive exploitation will 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 sources once again. In other words, the employed bee whose food source has been exhausted becomes a scout bee. The position of a food source in ABC corresponds to the possible 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), since each employed bee is associated with one and only one food source [16].

ABC algorithm as a population-based metaheuristic algorithm competes well with other population-based algorithms with an advantage of employing fewer control parameters [18], [19]. Due to its simplicity and ease of implementation, the ABC algorithm has captured much attention and has been applied to solve many practical optimization problems such as structural and concrete analyses [20], integer programming [21], leaf-constrained minimum spanning tree [22], digital IIR filter [23], real parameter control [24], generalized assignment problem [25], protein folding simulation [26], training of artificial neural networks [27], Numerical optimization [18], [28] constrained optimization problems [29], cluster based wireless sensor network routing [30], fuzzy clustering [31], reconfiguring distribution network [32], quadratic knapsack problem [33], job shop problem [34] and the lot-streaming flow shop scheduling problem [35].

The general format of the ABC algorithm as proposed by

Karaboga [16] is as follows:

- 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
  - Memorize the best food source found so far
- UNTIL (requirements are met)

IV. Artificial Bee Colony for CB-CTT

The ABC concept is described within the context of generating the best food source by artificial bees. Table 2 shows the equivalence between the optimization terms and the artificial bee colony terms.

Table 2. The optimization terms in the Artificial Bee context

ABC Terms	Optimization Terms
Employed or Onlooker Bees	$\leftrightarrow$ Solution
Nectar Amount	$\leftrightarrow$ Fitness of the solution
Repeated search for Food	$\leftrightarrow$ Iteration
Food Source Position	$\leftrightarrow$ Position of Solution
Food source	$\leftrightarrow$ Decision Variable
Best food source	$\leftrightarrow$ Best solution

The five main steps of ABC as adapted to CB-CTT, which is the form of pseudocode are described as follows:

Algorithm The ABC algorithm for CB-CTT

```
STEP 1  Initialize the CB-CTT and ABC parameters.
        Input the Data from CB-CTT instance.
        Formulate detailed knowledge CB-CTT: objective
        cost and solution representation.
        Define ABC parameter (MCN, Limit, Population,
        No of Employed, No of Onlooker).
STEP 2  Initialize the ABC Population (SN)
        Generate Food Source using Saturation
        Degree (SD) and store in the population SN
        SN = $x^1, x^2, ....., x^{SN}$
        Determine the Best food source in SN, $x^{best} \in SN$
        $f(x^{best}) \leq f(x^t), \forall t \in (1, .....SN)$
STEP 3  Generate a new food source.
        $x^{'} = \phi$ i.e. new food source
        for each employed bee
        $\rightarrow$ apply neighbourhood move
        $\rightarrow$ if the cost( move) $\leq$ cost (employed bee)
        $\rightarrow$ employed bee = neighbourhood move
        Determine the probability by using the cost
        select the onlookers to the sites
        $\rightarrow$ apply neighbourhood move as in employed bee
        $\rightarrow$ determine the fitness cost for the onlooker solution
        $\rightarrow$ if the cost(move) $\leq$ cost (employed bee solution)
        $\rightarrow$ employed bee solution = fitness of onlookers
        $\rightarrow$ select the scout bee by choosing the worst
        employed bee
STEP 4  Memorize Best food source.
        if($f(x') < f(x^{best})$) then
        $\rightarrow$ Replace with $x'$
        end if
STEP 5  Check the stop condition.
        while termination condition not met (MCN)
        $\rightarrow$ Repeat STEP 3 and STEP 4
        end while
```

Step 1. Initialize the CB-CTT problem and ABC Parameters
The solution to the CB-CTT problem is represented in Figure 1, where each course is scheduled to room(s) and period(s), subject to the number of lectures and minimum working days. ABC parameters such as SN, MCN, Limit are also initialized.

Step 2. Initialize the ABC Population (SN)
In step 2, the solutions are randomly generated using saturation degree as in [36]. These solutions are stored in the memory of ABC algorithm and the nectar amount or fitness is determined to obtain the best food source. It is worthy to note that the number of solutions is equal to the number of employed bees.

Step 3. Generate a New Food Source
In step 3, the new food source (timetabling solution) is produced by ABC algorithm using its three operators i.e employed, onlooker, and scout bees.

- **Employed Bees:** The Employed bee operator selects a food source i.e. feasible solution based on the objective function from the population of solutions. The employed bees work iteratively to produce a new solution on the neighborhood of this food source and the fitness of the solution is determined. This is done by selecting the courses randomly and assigning them to empty slots within the timetable solution.
- **Onlooker Bees:** The fitness of each food source is determined by the onlookers and the food source whose nectar amount or fitness is high is selected using tournament selection method; the onlooker exploits the solution applying similar criteria used by the employed bees. The onlooker abandons the food source if there is no further improvement and the employed bee associated with this food source becomes a scout.
- **The scout Bees:** These are known as the colony explorer navigates the search space for a new random solution and substitutes the one in the memory if it is better.

Step 4. Memorize Best Food Source
In step 4, the fitness value of the new food source found is evaluated by the ABC algorithm. If the fitness value of this new food source is better than the fitness of the best food source stored in the memory, the new food source replaces the best one in the memory.

Step 5. Stop Condition
Here, the ABC algorithm repeats steps 3 and 4 until the maximum number of cycles (MCN) parameter is met.

V. Computational Results and Discussion

In this section, the performance of ABC algorithm for solving the CB-CTT problem was evaluated using 21 data instances generated in ITC-2007. The details of the problem can be found in http://tabu.diegm.uniud.it/ctt/index.php. The proposed method is coded in Microsoft Visual C++ 6.0 under Windows Vista on an Intel Machine with CoreTM and a 2.66GHz processor and 2GB RAM. We ran the experiment 5 times for each problem instance for the purpose of statistical calculation. The overall penalty cost for each dataset was evaluated using the objective function formulation in equation (6), which adds up all the violations of soft constraints S1, S2, S3 and S4. The parameter setting for ABC algorithm is as shown in Table 3:

Table 3. ABC algorithm parameter setting

Parameter	Maximum Cycle	SN	Limit
Value	20000	90	100

Table 4 shows the experimental results of ABC algorithm and the best known results in literature; where the first column represents the best results produced by ABC, the second column shows the worst results of ABC while the last column shows the best results found in the literature, as reported in [8].

Table 4. Experimental Results of ABC

Problem instance	Best Result of ABC	Worst Result of ABC	Best in in Literature
Comp01	-	-	4
Comp02	312	391	20
Comp03	292	357	38
Comp04	193	224	18
Comp05	-	-	219
Comp06	336	392	16
Comp07	324	410	3
Comp08	218	246	20
Comp09	302	339	54
Comp10	274	371	2
Comp11	293	359	0
Comp12	-	-	239
Comp13	-	-	32
Comp14	236	295	27
Comp15	284	335	28
Comp16	281	330	16
Comp17	331	376	34
Comp18	196	217	34
Comp19	304	314	32
Comp20	372	457	11
Comp21	-	-	52

The result demonstrates that ABC algorithm could be tailored to solve CB-CTT problems. However, our approach is still not able to achieve results comparable with the best reported in the literature. An improvement (which is currently ongoing, with optimism) is required, in order to enhance the performance of the proposed method to produce desired outcomes better than those currently in use.

VI. Conclusion and Possible Future Directions

This paper presented the Artificial Bee Colony algorithm for tackling the CBC-TT problems. As the results have shown, the algorithm is capable of solving timetabling problems. Although the results produced by the algorithm in this study are presently not comparatively better than those already reported in the literature, this is initial adaptation of ABC algorithm. Further work will be necessary taking into consideration the of the problem.

Acknowledgment

This research was supported by the Universiti Sains Malaysia under Graduate Fellowship Scheme, 2011.

References

[1] 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.

[2] 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.

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

[4] 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.

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

[6] ——, "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.

[7] 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.

[8] 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.

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

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

[11] 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.

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

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

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*. Oxford University Press, USA, 1999.

[15] X. Yang, *Nature-inspired metaheuristic algorithms*. Luniver Press, 2010.

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

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] B. Akay and D. Karaboga, "Solving Integer Programming Problems by Using Artificial Bee Colony Algorithm," *AI* IA 2009: Emergent Perspectives in Artificial Intelligence*, pp. 355–364, 2009.

[22] A. Singh, "An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem," *Applied Soft Computing*, vol. 9, no. 2, pp. 625–631, 2009.

[23] 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.

[24] B. Akay and D. Karaboga, "Parameter tuning for the artificial bee colony algorithm," *Computational Collective Intelligence. Semantic Web, Social Networks and Multiagent Systems*, pp. 608–619, 2009.

[25] A. Baykasoglu, L. Ozbakir, and P. Tapkan, "Artificial bee colony algorithm and its application to generalized assignment problem," *Swarm Intelligence: Focus on Ant and Particle Swarm Optimization*, pp. 113–144, 2007.

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

[27] D. Karaboga and B. Akay, "Artificial Bee Colony (ABC) Algorithm on Training Artificial Neural Networks," in *Signal Processing and Communications Applications, 2007. SIU 2007. IEEE 15th*. IEEE, 2007, pp. 1–4.

[28] G. Zhu and S. Kwong, "Gbest-Guided Artificial Bee Colony Algorithm for Numerical Function Optimization," *Applied Mathematics and Computation*, 2010.

[29] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," *Foundations of Fuzzy Logic and Soft Computing*, pp. 789–798, 2007.

[30] D. Karaboga, S. Okdem, and C. Ozturk, "Cluster based wireless sensor network routings using Artificial Bee Colony Algorithm," in *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*. IEEE, 2010, pp. 1–5.

[31] D. Karaboga and C. Ozturk, "Fuzzy clustering with artificial bee colony algorithm," *Scientific Research and Essays*, vol. 5, no. 14, pp. 1899–1902, 2010.

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

[34] 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.

[35] 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.

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