Adaptation and Use of Artificial Bee Colony Algorithm to Solve Curriculum-based Course Time-Tabling Problem

Saeid Agahian
Computer Engineering Department
Karadeniz Technical University
Trabzon, Turkey
saeid@ktu.edu.tr

Huseyin Pehlivan
Computer Engineering Department
Karadeniz Technical University
Trabzon, Turkey
pehlivan@ktu.edu.tr

Rahim Dehkharghani
Computer Science and Engineering Program
Sabanci University
Istanbul, Turkey
rdehkharghani@sabanciuniv.edu

Abstract—Curriculum-Based university Course Time-Tabling, CB-CTT, a known scheduling problem. We adapted a new swarm intelligence approach, identified as MABC based on the Artificial Bee Colony (ABC) to solve the CB-CTT. The approach consists of two steps: first, a feasible solution of the problem is constructed, which satisfies only the hard constraints; and then, the soft constraints are attempted to be satisfied. MABC could satisfy the hard constraints of the problem for all datasets of the ITC-2007 track 3, a benchmark dataset for the CB-CTT. The penalty of the achieved solutions by MABC is comparable to the related work in the literature that used the ABC for solving the CB-CTT.

Keywords—Curriculum; course time-tabling; Artificial Bee Colony Algorithm; Meta Heuristic Methods;

I. INTRODUCTION

In our daily life, there are many scheduling problems such as time-tabling. “The allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way to satisfy as nearly as possible a set of desirable objectives” [1] is the formal definition of time-tabling. In time-tabling problems, hard constraints must be satisfied, while the more satisfied soft constraints, the more desirable the achieved solution is [2]. Educational time-tabling is divided into three main sub-problems: school, course and exam time-tabling [3]. We focused on course time-tabling in this paper. We applied the ABC [4] to the CB-CTT problem. An advantage of the ABC is its ability in local and global search in a big space using less number of control parameters compared to other approaches such as the PSO and Genetic algorithm. Grey and Johanson [5] proved the NP-completeness of this problem. Meta heuristic methods are appropriate alternatives for solving this problem.

Meta-heuristic Methods start with a number of randomly initiated solutions and then apply search algorithms to the current state of the problem to obtain a better solution. Several meta heuristic methods are basically derived from the nature such as ABC. Lewis [6] divides the meta heuristic methods applied on university course time-tabling problem into three main groups: one stage optimization algorithm, two stage optimization algorithms and algorithms that allow relaxations.

Designing a standard time-table and curriculum as the benchmark can help competitors in of CB-CTT. One such benchmark dataset was proposed by Socha [7], [8] including eleven datasets. Another one has been proposed by Meta heuristics Network [9] in the ITC-2002 [10]. This dataset was generated randomly and updated in the ITC-2007 [11]. It was created based on real data from the University of Udine in Italy. The current paper proposes an algorithm based on the ABC from the second group mentioned above (two stage optimization algorithms), using the benchmark dataset of the ITC-2007 track 3. The goal of this paper is to satisfy both hard and soft constraints. We were able to achieve at least one feasible solution for each dataset. In the second step, the proposed algorithm could achieve fairly well results compared with the literature. The most recent competition held on the CB-CTT is the ITC 2007. The current work relates only to the track 3 of this competition. Our results could place among five smallest penalties obtained by five winners of this competition. Our contribution can be summarized as: unlike other related work, we used the ABC for both steps of the CB-CTT. The algorithm proposed for neighbor searching in this paper is novel; and the local search algorithm has been adapted to the ABC to improve the results.

II. THE ITC-2007, CURRICULUM BASED TYPE

We attempt to dedicate the lectures of all courses to available time-slots and compatible rooms during the week. We also consider the curricula of the university to prevent any conflict between the courses in the same curriculum. Here are some definitions:

• Course: available lessons for students.
• Lecture: each course consists of a fixed number of lectures.
• Day and time-slot: each week consists of five or six working days and each day is divided into five or six time-slots.
• Period: it is the pair of [day, time].
• Room: these are actually the classrooms.
- Curricula: each curriculum consists of a set of courses, each pair of which may have students in common.

Soft constraints are defined based on each curriculum. We collected input data for the CB-CTT problem in four distinct data structures:
- \( C_i \), the \( i \)th course with five fields: course name, teacher name, number of lectures \( (l_i) \), the minimum working days \( (mwd) \) and the number of students.
- \( CR_i \), the \( i \)th curriculum with the basic fields: curriculum name, course name and number of courses
- \( r_i \), the \( i \)th room with two fields: room name and room capacity.
- \( un_i \), the tuple \( (d, t) \), where \( d \): day and \( t \): time-slot are unavailable for the lectures of the course \( C_i \).

A candidate solution is represented by three-dimensional variables in a certain period, none of the lectures of the course can be assigned to it.

\[
\sum_{i=1..p, j=1..h, k=1..m} L(x_{i,j,k} == c \in C) \text{ and } (i, j) \in un_c = 0
\]

The soft constraints are listed here:
- **Room Capacity**: a lecture should be assigned to a room that has the capacity greater than or equal to the number of students in the lecture. For each extra student, the penalty of one is added.

\[
\forall x_{i,j,k} = c \in X, p_1(x_{i,j,k}) =
\begin{cases} 
(c_z \to std) - (r_k \to cap) & \text{if } (c_z \to std) > (r_k \to cap) \\
0 & \text{otherwise}
\end{cases}
\]

- **Minimum Working Days (MWD)**: all lectures of a course should be spread into the given minimum number of days. For values less than the MWD, the penalty of five (for each day) is added. The function \( wd(c) \) is the number of working days that \( c \) has been scheduled in.

\[
\forall c_z \in C, wd(c_z) =
\sum_{i=1..p} L(x_{i,j,k} == c_z), j = 1..h, k = 1..m
\]

\[
p_2(c_z) =
\begin{cases} 
(c_z \to mwd) - wd(c_z) & \text{if } wd(c_z) < (c_z \to mwd) \\
0 & \text{otherwise}
\end{cases}
\]

- **Curriculum Compactness**: Adjacent lectures in timetable, are supposed to be from the same curriculum. For each lecture not adjacent to one from the same curriculum, the penalty of two is added. Function \( X\text{OR} \) returns 0, if its operands are equal, or 1 otherwise.

\[
\forall cr_z \in CR, p_3(cr_z) =
\sum_{i=1..p, j=1..h-1} [L(cr_z \in cur(x_{i,j,k}|k = 1..m)) X\text{OR} (cr_z \in cur(x_{i,j+1,k}|k = 1..m))]
\]

- **Room Stability**: the lectures of a course should be scheduled in the same room. For each additional room, the penalty of one is added.

\[
\forall c_z \in C, p_4(c_z) = \text{length}\{k|x_{i,j,k} == c_z, i = 1..p, j = 1..h, k = 1..m\} - 1
\]

After computing the penalty of violating the soft constraints, the penalty values are combined using the following formula, where the digits inside the sigma functions are the coefficients of soft constraints.

\[
P(X) = \sum_{i,j,k} P_1(x_{i,j,k}) + \sum_{c_z \in C} 5* P_2(c_z) + \sum_{cr \in CR} 3* P_3(cr_z) + \sum_{c_z \in C} 3* P_4(c_z)
\]
There are 21 datasets in a benchmark database. There is at least one feasible solution for each dataset [12]. Details of all of datasets are available in [13]. The ITC-2007 has announced time limit (600 seconds) for achieving the solutions after the second step of CBCTT, which is 483 seconds for our PC based on the evaluation program of the ITC-2007.

III. RELATED WORKS

There exist several categorizations of algorithms for solving the CB-CTT. One group is based on the sequential methods such as mathematical programming and graph coloring. Integer programming is a member of this group, which has been used in recent works such as [14], [15], [16], [17], [18], [19]. More recent work in this group have been accomplished by Lach and Lubbecke [20], [21]. A subgroup within this group is mixed integer programming, which was used by [22].

Other work from this group are based on graph coloring [6], [23]. In this approach, each node represents a lecture. An edge is created between two nodes if there is a conflict between them. Muller[24] applied constraint based approaches.

In the ITC-2007 competition, five winners are Muller [25] with Constraint based algorithm, Lu and Hao [26] with adaptive Tabu Search, Atsuta et. al. [27] with Constraint satisfaction problem (CSP), Geiger [28] with Stochastic neighbourhood search and Clark et al [29] with Repair based heuristic search. After 2007, other researchers suggested other algorithms for solving this problem. De Cesco et al [30] and Salwani Abdullah et al [31] exemplify them, which used the methodology of Chiarandini et. al.[32] was used in the first step, but a new approach was employed only for the second step of the CB-CTT. The most similar work to ours has been done by Bolaji et. al[identified as IABC] [33]. The authors employed a two-phase mechanism to deal with the problem. In the first phase, the lectures are sorted by the Saturation degree algorithm; then they are assigned to the available rooms using a back-tracking algorithm. The second phase, on the other hand, tries to satisfy the soft constraints by typical swap (changing the content of two elements in solution) and move (moving the content of one element to an empty one in the solution) operations, as well as the original ABC algorithm (Section VI).

Section VI presents the comparison of our work with the winners of the ITC-2007 and other related works.

IV. ARTIFICIAL BEE COLONY

In the ABC, bees are divided into employed, onlooker and scout bees. The employed bees search for food sources and bring the collected information from all sources back to the hive. The onlooker bees choose one of the best food sources based on the dance style of the employed bees. Scout bees randomly search the environment around the hive for food when necessary.

The ABC has three parameters: 1) SN: number of food sources, 2) limit: maximum number of tries for replacing the food source with a new one and 3) D: number of optimization parameters. The ABC is divided into three main phases after initialization: the employed phase, the onlooker phase and the scout phase. These phases are explained in [4] with more detail.

V. APPLYING THE ABC TO THE CB-CTT

In order to represent a time tabling problem as a permutation problem, we used a three dimensional matrix: day * time-slot * room. Each matrix is a potential final solution.

A. First Step of the CB-CTT

The process of adapting each matrix to a food source in the ABC is as follows:

Initialization phase: the lectures are distributed into rooms and periods randomly. First available position in the matrix is dedicated to a lecture, if this assignment satisfies at least three hard constraints: lectures, Room occupancy and availability.

Employed phase: employed bees search and find the neighbours of previously found solutions and measure their optimality. A neighbour of a solution is a solution that is derived from the main one without violating the hard constraints. There are several algorithms for neighbour search explained in [34]. Our approach is as follows: First, the room is considered as the dimension for changing a solution to obtain the new one. One room in the current solution, is randomly chosen. Another solution is also randomly chosen, in order to locate the neighbour in its direction. We are interested in getting closer to the chosen solution by changing the current one, which is approved only if no hard constraint is violated. Afterwards, the fitness function is executed on the newly constructed neighbour; if the fitness value of the new solution is greater than the previous one, the previous solution is replaced by the new one. The fitness function (with time complexity of $O(nm^2)$) is the inversion of the penalty ($\frac{1}{n}$) of violating the soft constraints. The process explained above is repeated for all solutions. At the end, a probability value is calculated for each solution, based on its fitness value:

$$p_i = \frac{f_i}{\sum_{i \in food sources} f_i}$$

$f_i$ is the output of fitness function for a solution. A probability value is assigned to each solution based on its optimality in the employed phase.

Onlooker phase: here, the foods are chosen with replacement for N times (number of onlooker bees) by Roulette wheel algorithm based on the probability values achieved from the previous phase. This phase is similar to previous one with one differences: only some of the solutions are...
searched to find the neighbours.

**Scout phase:** if no better neighbour is found after N (limit) tries in previous phases, current solution should be replaced by a randomly generated one. The difference of the MABC with the original ABC is that we do not generate a new solution from the scratch but we change only 30 percent of it. We repeat the process of generating the changed solution until a solution satisfying all hard constraints is obtained:

**Initialization Phase**

1) Set the parameters $SN$ (number of sources) and $limit$ and then read dataset from input file. Then set parameter $D$.
2) Initialize the food sources and evaluate each food source according to fitness function.
3) Construct conflict matrix based on the curricula of the problem.
   a) Generate a random permutation of courses and store in $C$
4) Repeat $SN$ times
   a) Generate a food, initialized by -1.
   b) Repeat for each element of $C$
      i) Repeat for each lecture of $c_i$ (max 100 times).
         A) Generate random triple (day, time, room).
         B) If $c_i$ can be assigned to the triple without violating three hard constraints (all except the conflict constraint), then assign it and break.
         C) If the number of iterations is greater than 100, go to Step 4.a.
5) Evaluate foods according to violation of hard constraints and save fitness value of each food.

**Employed Phase**

Repeat for each food $f$

1) Generate a random number as the neighbour of $f$ and a random room number.
2) Repeat for each element of the room in $f$ (start element is chosen randomly).
   a) Assume $x_i$ is the content of $f$ in chosen room and $y_i$ is the content of the neighbour food in the same coordinate.
   b) Search the food for each $y_i$ that can be exchanged by $x_i$, if found then swap $x_i$ and $y_i$ in the food.
3) If the new solution is better than $f$ based on the soft constraints penalty, replace it with $f$ and save fitness value.

**Onlooker Phase**

Calculate the probability of foods. Repeat for each food

1) Select a food for onlooker bees using the roulette wheel algorithm according to the probability.
2) Produce the neighbour of the food as in the employed phase.

3) If the solution is better (with less of a penalty) than $f$, replace the older solution with this one and save the fitness value.

**Scout phase**

- If $f$ has not been improved during the limit times of iteration, change 30 % of $f$ and abandon it; then save its fitness value.
- Memorize the best food and assign its fitness value to the global-min.
- Repeat all the steps after the employed phase until global-min is greater than zero (max 2000 times). If the repeat loop counter raised to 2000, then go to the initialization phase.

**B. Second Step of the CB-CTT**

There are a few differences between the phases of two steps which are mentioned below:

**Initialization phase**: all stages of the first step in the CB-CTT correspond to the initialization phase of the second step.

**Employed phase**: in this phase, the vicinity of each solution is searched to find other solutions. The following process is repeated for all foods:

- A solution (destination) and a dimension (room number) is randomly chosen.
- The source solution should be similar to destination, in chosen direction, based on the following algorithm:

Each lecture in the current element needs to be the same as its equivalent element in the same period in destination. For example, $l_i$ is the lecture assigned to day 2 and time-slot 3 of destination but $l_j$ resides in equivalent day and time-slot of the source. In order to replace $l_i$ with $l_j$, the other indexes including lectures of course $c$, where $l_i \in c$, are searched in source solution; if they could be found in period $k$, the content of period $[2, 3]$ is exchanged with the content of period $k$. This change is approved only if the changed solution is still feasible and its penalty for soft constraints is not greater than the penalty once it was entered to neighbour search process. Then, local search is applied on the newly obtained solution to make it better, if possible. Local search works as follows:

Two triples of (period, lecture, room) are chosen randomly. If their content can be exchanged without violating the hard constraints and also with decreasing the penalty of soft constraints, they will be exchanged, otherwise, two other triples are chosen. This process will be repeated until two triples can be exchanged for maximum of fifty tries. At the end of this phase, a probability value is generated for each resource.

**Onlooker phase**: here, foods are chosen with replacement for N times, using Roulette Wheel algorithm, based on the probability values calculated in the previous phase.
It means that those foods with higher quality are chosen more than the others. After choosing a solution, the same neighbour searching process in employed phase is repeated here. The difference between the onlooker and employed phases is not to use local search algorithm in onlooker phase. At the end of each phase, if solution could not be improved, counter increments. This update of counters for each solution will be used in scout phase.

**Scout phase:** If the number of iterations that food $f_i$ could not be improved, is greater than the limit parameter, current food would be changed at most for 30 percent, in at most 1000 tries. The difference of our algorithm in this phase with scout phase of the original ABC is that we treat non-optimal solutions and the optimal one differently. We could be able to achieve better results when the optimum solution was not violated. If the solution is optimum, no exchange in solution matrix is allowed to increase the soft constraints penalty; but in other cases, only satisfying two soft constraints is enough: room capacity and room stability.

VI. Evaluation and Comparison

We evaluated the proposed approach with the ABC parameters of limit=15, SN=16/2=8 and D as the number of rooms, which is dataset dependent. The results have been obtained from a PC running under Windows 7 with an Intel(R) Core(TM) 2 Duo 1.8 GHz CPU and 2 Gigabytes RAM. Our focus in this paper is on the second step of the CB-CTT. The penalty of violating the soft constraints in the second step is presented in Table I. Numbers 1 through 5 stand for five winners in ITC-2007 track 3 and the IABC is the most recent work similar to ours for solving the CB-CTT. In most of the datasets, our results could place between the third and forth winners. After 2007, a few research work attempted to solve the CB-CTT with the ABC. The best results with lowest penalty -among them were achieved by IABC. The differences (or advantages) of the MABC with the IABC can be summarized as 1) using local search for neighbor searching in the MABC and 2) using the ABC for solving both steps of the CB-CTT problem in MABC, while the IABC has used the ABC only for the second step.

VII. Conclusion

Curriculum based university course time-tableing has two steps: in the first step, one tries to find a feasible solution that satisfies hard constraints while the second step is concerned with satisfying both hard and soft constraints in pre-found feasible solutions. This paper is related to both steps, where we tried to solve it by a modified version of the ABC algorithm. The algorithm could solve the CB-CTT problem and find at least one feasible solution for all datasets. Besides, the penalty of violating soft constraints in our work is comparable to the results of state of the art related work.

**References**


Table I

<table>
<thead>
<tr>
<th>Systems</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>IABC Average</th>
<th>MABC Best</th>
<th>MABC Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>comp01</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>24</td>
<td>5</td>
<td>6.1</td>
</tr>
<tr>
<td>comp02</td>
<td>51</td>
<td>55</td>
<td>50</td>
<td>111</td>
<td>111</td>
<td>299</td>
<td>86</td>
<td>115.5</td>
</tr>
<tr>
<td>comp03</td>
<td>84</td>
<td>71</td>
<td>128</td>
<td>119</td>
<td>166</td>
<td>57</td>
<td>77</td>
<td>67.3</td>
</tr>
<tr>
<td>comp04</td>
<td>37</td>
<td>43</td>
<td>35</td>
<td>72</td>
<td>72</td>
<td>166</td>
<td>57</td>
<td>67.3</td>
</tr>
<tr>
<td>comp05</td>
<td>330</td>
<td>309</td>
<td>312</td>
<td>410</td>
<td>426</td>
<td>456</td>
<td>377</td>
<td>405.9</td>
</tr>
<tr>
<td>comp06</td>
<td>48</td>
<td>53</td>
<td>69</td>
<td>100</td>
<td>130</td>
<td>255</td>
<td>87</td>
<td>94.3</td>
</tr>
<tr>
<td>comp07</td>
<td>20</td>
<td>28</td>
<td>57</td>
<td>110</td>
<td>253</td>
<td>61</td>
<td>73</td>
<td>75.3</td>
</tr>
<tr>
<td>comp08</td>
<td>41</td>
<td>49</td>
<td>77</td>
<td>83</td>
<td>173</td>
<td>60</td>
<td>70</td>
<td>70</td>
</tr>
<tr>
<td>comp09</td>
<td>109</td>
<td>105</td>
<td>110</td>
<td>139</td>
<td>271</td>
<td>127</td>
<td>132</td>
<td>132.4</td>
</tr>
<tr>
<td>comp10</td>
<td>16</td>
<td>21</td>
<td>71</td>
<td>85</td>
<td>239</td>
<td>51</td>
<td>51</td>
<td>64.2</td>
</tr>
<tr>
<td>comp11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>220</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>comp12</td>
<td>333</td>
<td>343</td>
<td>442</td>
<td>408</td>
<td>751</td>
<td>397</td>
<td>404.8</td>
<td></td>
</tr>
<tr>
<td>comp13</td>
<td>66</td>
<td>73</td>
<td>622</td>
<td>113</td>
<td>214</td>
<td>90</td>
<td>97.9</td>
<td></td>
</tr>
<tr>
<td>comp14</td>
<td>59</td>
<td>57</td>
<td>90</td>
<td>84</td>
<td>221</td>
<td>77</td>
<td>85</td>
<td></td>
</tr>
<tr>
<td>comp15</td>
<td>84</td>
<td>71</td>
<td>128</td>
<td>119</td>
<td>238</td>
<td>92</td>
<td>106.7</td>
<td></td>
</tr>
<tr>
<td>comp16</td>
<td>34</td>
<td>39</td>
<td>81</td>
<td>152</td>
<td>280</td>
<td>110</td>
<td>122.1</td>
<td></td>
</tr>
<tr>
<td>comp17</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>84</td>
<td>236</td>
<td>83</td>
<td>99.4</td>
<td></td>
</tr>
<tr>
<td>comp18</td>
<td>83</td>
<td>69</td>
<td>116</td>
<td>110</td>
<td>173</td>
<td>97</td>
<td>121.8</td>
<td></td>
</tr>
<tr>
<td>comp19</td>
<td>62</td>
<td>65</td>
<td>107</td>
<td>111</td>
<td>276</td>
<td>82</td>
<td>92.6</td>
<td></td>
</tr>
<tr>
<td>comp20</td>
<td>27</td>
<td>47</td>
<td>61</td>
<td>88</td>
<td>244</td>
<td>77</td>
<td>89.7</td>
<td></td>
</tr>
<tr>
<td>comp21</td>
<td>103</td>
<td>106</td>
<td>123</td>
<td>174</td>
<td>364</td>
<td>74</td>
<td>90.5</td>
<td></td>
</tr>
</tbody>
</table>


