Scheduling with Divide and Conquer Genetics – A Model and Implementation

A. S. Elhassan#1, L. Al Zubaidi#2, S. Al-Huwaider#3
College of Computer Engineering & Science
Prince Mohammad Bin Fahd University
Al-Khobar, Kingdom of Saudi Arabia
#1aelhassan@pmu.edu.sa
#2lalzubaidi@pmu.edu.sa
#3201301805@pmu.edu.sa

Abstract - The Final Exams Scheduling Problem (FESP) that has multiple constraints and objectives relating to student and professor conflicts as well as availability of timeslots is a known NP-Hard problem. This has been addressed using tweaked computational algorithms that are based on the Genetic approach. The approach was implemented using C# with an Oracle database of the scheduled student classes for one semester in a small size university with 16000 student-section instances that were aggregated to 486 unique section offerings requiring a non-conflicting schedule. Tests showed positive results that were improved with the added heuristics to the standard Genetic Approach.

Keywords - Crossover, Fitness, Genetic Algorithm, Mutation, Timeslots.

I. INTRODUCTION

One of the most important functional areas within academic schools, colleges and universities is course and lecture scheduling to classrooms within semesters for student learning as well as the scheduling and timetabling of final exams for all students during a defined time period for assessment. This is a combinatorial optimization (NP-hard) problem [1] with no known deterministic polynomial time algorithm for solving it [2, 3, 4]. The same algorithms and techniques that apply in this area are also applicable in other domains like airline crew scheduling or cloud computing task autonomy scheduling and many other high complexity problems wherein the requirement is to formulate solutions for combinational optimization problems with multiple constraints and objectives [5, 6].

Although the constraints related to course section scheduling tend to include many common factors they may vary from university to university or depending on region or locale; for example, in co-ed environments the requirement of normally for one schedule that caters for the whole institute, however, in other environments where gender segregation occurs there is a need for two or more schedules; this can be made more complex if resource (time) sharing occurs, for example, same labs, lecture halls or instructors. FESP is one such example where instructors teach both male and female classes and it is often a requirement that the instructor be available in his/her exam session and hence, instructor conflicts are a factor that is as important as student conflicts.

This article is structured as follows: Background section discusses the various algorithms and solutions to the scheduling problem, the Problem Description and Formulation section analyses the typical set of requirements and constraints, the Design and Implementation section explains the algorithm that is proposed in this article, the results of the test runs on the 486 nodes database are presented in the Experimental Results section and the final section is Conclusions and Future Work.

II. BACKGROUND

There is quite a large number of theoretical and applied research methodologies in this area with more ideas continuing to emerge to take advantage of advancements in the theory and physical processing power of computing. The most common solutions can be generally categorized as follows [1,8, 9]:

- **Graph Coloring**
- **Population Based Solutions** e.g., Genetic Algorithms (GA) and Ant Colony (AC) [24]
- **Meta-Heuristics** e.g., Tabu Search (TS) and Simulated Annealing (SA) [15, 16]
- **Constraint-Based** Solutions [25]
- **Variable Neighborhood Search** (VNS)
- **Combination/Hybrid** based on a combination of two or more of the above or by modifying the main process within [14]
In terms of the search scope we can classify solutions into Local-Area-based algorithms and Population (Wider-Area) based algorithms [2]. Local Area Search algorithms include SA, VNS, TS, and many more; these select a subset of the solution space and move in one direction without performing a wider scan of the whole search space, hence although they tend to be faster; they focus on exploitation rather than exploration.

The (Wider) Population search algorithms that include Genetic and Evolutionary algorithms (GA/EAs), particle swarm optimization, ant-colony optimization, artificial immune system as well as other AI based systems start with a number of solutions and refine repeatedly until an acceptable optimal solution is found from the whole search space, hence, they operate in global-area-based fashion but are subject to resource, speed and local optima issues.

This research is based on a graduation project at a small university of 5000 students and is a continuation of the work in [9] where first fit and graph coloring approaches were tested and compared with novel course node sorting heuristics by adjacency. In this article we are attempting to produce feasible timetables for final exams using the Genetic Approach (GA) for separate timeslot-days in an exclusive fashion that treats each timeslot-day as sub-timetableting problem. The choice of GA has been established since the 1970s as a relatively optimal approach with applied solutions in the 1990s in Japanese schools for textbook arrangement [10].

### III. Problem Description and Formulation

FESP main requirement is to generate a final exam schedule for 2 gender-segregated campuses with shared instructors, the approach is based on the following “hard” constraints:

- C1: No Student Conflicts: A student cannot be scheduled in 2 or more exam sessions simultaneously
- C2: No Instructor Conflicts: An instructor cannot be scheduled in 2 or more exam sessions simultaneously
- C3: There are 10 exam days
- C4: Each day has 2 exam periods: (0900 and 1200)
- C5: Female students have to be scheduled in the female campus
- C6: Male students have to be scheduled in the male campus
- C7: A classroom cannot be used for mixed (topic) exams
- C8: No exams on Weekends

There are some other “soft” constraints to satisfy, as follows:

- F1: The schedule should fill all the available rooms, every day.

The registrar’s office will provide 2 (per campus) rosters of the student-section-instructor offerings in the following format in table 1 (with examples):

<table>
<thead>
<tr>
<th>Student ID</th>
<th>Student Name</th>
<th>Course-Section</th>
<th>Instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013011234</td>
<td>Chris Pitt</td>
<td>CSC1_201</td>
<td>Stevens A.</td>
</tr>
<tr>
<td>2012120123</td>
<td>Brad James</td>
<td>CSC2_101</td>
<td>Nicholl S.</td>
</tr>
<tr>
<td>2013011234</td>
<td>Chris Pitt</td>
<td>HST1_201</td>
<td>Amanda G.</td>
</tr>
<tr>
<td>2014120129</td>
<td>Allan Smith</td>
<td>GEG1_101</td>
<td>Nicholl S.</td>
</tr>
</tbody>
</table>

Hence, constraint C1 dictates that Chris Pitt prevents CSC1_201 and HST1_201 from being scheduled simultaneously and constraint C2 dictates that CSC2_101 and GEG1_101 cannot be scheduled simultaneously and so on.

The end schedule needs to be of the structure shown in table2, which shows that there are 86 unscheduled courses, this is not a failure because there are 10 schedule days and 2 timeslots per day with 20 exam classrooms, and hence the maximum number of courses that can be scheduled is $10 \times 2 \times 20 = 400$.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Room1</th>
<th>Room2</th>
<th>Days</th>
<th>406</th>
</tr>
</thead>
<tbody>
<tr>
<td>08-May-17</td>
<td>9:00</td>
<td>PHD1122_208</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>08-May-17</td>
<td>12:00</td>
<td>PHD1312_202</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>09-May-17</td>
<td>9:00</td>
<td>PHD1112_205</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>09-May-17</td>
<td>12:00</td>
<td>PHD1312_210</td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

The elements of a typical FESP model are as follows [11]: Let $U$ be the set of all student ID’s who are registered in some course-section this semester and $S$ is the set of all course-section offerings, as shown in table 1, above. The adjacency matrix $M$ with vertex set $|S| \times |S|$ maps the elements of $S$ to themselves such that element $M_{ij}$ is set to “1” when there is (at least one) a common student between Course-Section $S_i$ and Course-Section $S_j$, and “0” when there is no common students/instructors, and is also set to “0” when $i = j$, i.e., $M$ is symmetric on the main diagonal, which is set to zero values. Table 3, below shows an adjacency matrix for a 12 node (course-section) schedule.
Because Oracle is used to construct the adjacency matrix as a data table with column and row headings, the order of the rows and columns is insignificant while the C# application is referring to the data but is preserved correctly when using Visual Studio .Net arrays to model the data, hence, the adjacency matrix in Table 4 is 100% identical to the one in Table 3.

Table 4. Adjacency Matrix for 12 Course-Sections in Table 3

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
<th>C7</th>
<th>C8</th>
<th>C9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

IV. FESP DESIGN AND IMPLEMENTATION

The Application Interface, shown in figure 1 has been designed to support the requirements C1 to C8 in addition to soft constraint S1 (in the case of the example run in Table 2 above).

Figure 1. FESP Application Interface

The process of generating groups of courses consists of several phases each with several steps; the process of the transformation from the registrar’s class roster (Table 1) to the adjacency matrix (Table 3) is assumed to be sufficiently clear to the reader. The aim of the FESP solution and application is to generate compatible section groups \( \{g_1, g_2, \ldots, g_n\} \in G \) such that each group size is as close to the number of rooms \( |R| \) as possible, the approach thus divides the complex full scheduling problem into a subset of scheduling problems each for one group of \( |R|/\text{section/rooms} \) [12], this is illustrated in figure 2a and 2b.

The main features of this process that have been introduced here are:

(i) Seeded genome generations for a root (DNA), as shown in the “Create Genomes for Root” in Fig 2a, and

(ii) Carry 2 fittest genomes from current generation when creating next one, as shown in “Select 2 Fittest Genomes from Generation (i)” in Fig 2b.
Compatibility/fitness for a group $g_x = \{s_{x1}, s_{x2}, s_{x3}, \ldots, s_{xn}\}$ assigns a percentage value indicating the rate of student/instructor conflicts between the sections of $g_x$ determined as follows:

Algorithm 4.1: Pseudo-code: Group_Fitness

Input(s):
- $g_x = \{s_{x1}, s_{x2}, s_{x3}, \ldots, s_{xn}\}$ //group of proposed sections
- $M$ //adjacency matrix

Output(s):
- double fit //percentage of fitness (compatibility) between elements of $g_x$

Begin

double $f = 0$;
int $n = |g_x|$;
for (int $i = 0 ; i < n; i++$)
{
    for (int $j = i + 1; j < n; j++$)
    {
        if ($M_{ij} == 1$) $f++$;
    }
}
Return $100 \times ((0.5(n(n - 1)) - f) / ((0.5(n(n - 1))))$;

End

V. EXPERIMENTAL RESULTS

The algorithm alterations have produced positive impact on the overall scheduling performance results for a 500 nodes database as shown in the results in Fig. 3 for runs with a seed root and NOT carrying over the 2 best genomes from one generation to another, Population Size: 8, Target Group Size: 20, Number of Generations: 20, Crossover Rate: 40% and mutation Rate: 60% shown in blue (P8 Series). Compared to runs with a seed root and also NOT carrying over the 2 best genomes from one generation to another, Population Size: 40, Target Group Size: 20, Number of Generations: 80, Crossover Rate: 20% and mutation Rate: 80% shown in orange (P40 Series).
Running the same algorithm with the same 500 node database WITH carrying over the 2 best genomes from one generation to another and identical parameters to the previous results in Fig 3 produces further performance gains as shown in Fig. 4, with the improved performance with best 2 fit genomes carried over from one generation to another shown in blue (series P40').

The full experimental results are shown in table 6 with the first and second groups not applying the carry best 2 fit parents forward and the last group applying this step. Regardless of the results shown, in all cases, the hard constraints (C1 to C8) are always satisfied by all solutions and soft constraint F1 is the one that is compromised; all groups of course nodes that are generated by the algorithm are 100% compatible/fit in relation to each other for both instructor and student and as per the fitness algorithm (4.1). Each combination of heuristics was tested with 8 application runs over the same 486 node dataset.

VI. CONCLUSIONS & FUTURE WORK

The NP-Complete problem of FESP has several methodologies and implementation combinations to resolve it [12], taking the Genetic Algorithm [13] approach and modifying its population generation heuristics we have implemented a solution application that works in conjunction with database engines to generate marginal improvements to other solutions, the potential for this approach is positive if combined with other heuristics [9, 16]. The test database consists of 486 nodes and further tests with 500-1000 nodes database will be conducted as extensions to this capstone student project research within a small academic college.
ACKNOWLEDGMENT

The authors would like to acknowledge the contribution of Prince Mohammad Bin Fahd University, KSA for providing the support to conduct this research work.

REFERENCES


[5] Florin Pop; Valentin Cristea; Nik Bessis; Stelios Sotiriadis, "Reputation Guided Genetic Scheduling Algorithm for Independent Tasks in Inter-clouds Environments”. 27th International Conference on Advanced Information Networking and Applications Workshops 2013, Pages: 772 - 776, DOI: 10.1109/WAINA.2013.206


[9] A. Elhassan, “Graph-Coloring for Course Scheduling to Time-Room Slots - A Comparative Analysis Based on Course Selection Order”, the Third International Conference on e-

Technologies and Networks for Development (ICcND2014) Lebanese University - Beirut, April 29, to May 1 2014


