Memetic Algorithms for Solving University Course Timetabling Problem

Morteza Alinia Ahandani
Department of electrical engineering
Islamic Azad University, Langaroud branch
Langaroud, Iran
alinia@iaul.ac.ir

Mohammad Taghi Vakil Baghmisheh
Department of control engineering
University of Tabriz
Tabriz, Iran
mvakil@tabrizu.ac.ir

Abstract— In this paper we propose two memetic algorithms for the university course timetabling problem. First by using graph coloring heuristics a new crossover method is proposed. Then to hybridize the genetic algorithm, a local search method with a mechanism similar to hill climbing is defined. The proposed memetic algorithms are applied on some datasets and their performance are compared with each other. The obtained results demonstrate that the first memetic algorithm has a better performance than the second one. Also a saturation degree heuristic is utilized in the crossover which improves the performances of the memetic algorithms. Comparison of the obtained results with the results reported in the literature demonstrates the comparability of our proposed algorithms with already existing algorithms.

Keywords- course timetabling; memetic algorithm; graph colouring

I. INTRODUCTION

An interesting subclass of scheduling problems is timetabling problems, which may be highly constrained and difficult to solve. A general timetabling problem consists of assigning a number of events (exams, courses, meetings, etc) into a limited number of timeslots, while minimizing violations of a set of constraints. Different timetabling problems have different constraints. Generally, constraints are classified into two classes: hard constraints and soft constraints. Hard constraints have to be satisfied under any circumstances, and solutions with no violation of hard constraints are called feasible solutions. Other constraints are called soft constraints, satisfying which are desirable but not essential. Indeed, in most of the real world problems, due to their complexity, it would be impossible to satisfy all of the soft constraints. Thus to generate some feasible solutions, it would be necessary to violate some of them. Violations of soft constraints are utilized in the cost function to evaluate the quality of the obtained solutions [1]. Solving timetabling problems includes both time-based planning and combinatorial optimization which necessitates utilizing a stochastic search algorithm such as evolutionary algorithms (EAs) along with a heuristic method such as sequential graph coloring heuristics.

Graph coloring have been widely used to solve the timetabling problems. In fact, conventional computer-based automated timetabling methods concern themselves with finding the shortest timetable that satisfies all hard constraints, through a sequential graph coloring heuristics, and pay little attention for optimizing soft constraints. For example the reported results in [2] demonstrate that sequential heuristics had proved to be very efficient when incorporating a backtracking procedure. Reference [3] employs a heuristic procedure without backtracking but incorporates a random element and [4] proposes a fuzzy heuristic ordering.

This research work attempts to combine two different approaches. The first approach is a heuristic approach involving graph coloring methods which usually lead to satisfactory and feasible solutions. The second approach is known as EAs which usually lead to near optimal solutions and can be used as an optimization approach on soft constraints.

Genetic algorithms (GAs) have been widely used for scheduling problems, too. Reference [5] describes the use of a GA to solve timetabling problems and reference [6] implements a university examination timetabling method based on a GA for the Middle East Technical University. The results reported by applying the GA on timetabling show that the original GA cannot produce a good quality solution [7, 8]. As a result, often a combination of GA with other algorithms is used to improve the quality of obtained solutions. In this research work, the modified GA is hybridized with a novel local search method.

The rest of the paper is organized as follows. In section 2, course timetabling problem are briefly explained. In section 3, the graph coloring heuristics are described. In section 4, the modified GA and the novel local search algorithm, and the method of their hybridizing are explained. The simulation results are presented and analyzed in section 5. Section 6 concludes the paper.

II. COURSE TIMETABLING PROBLEM

University course timetabling consists of a set of courses (lectures, laboratories, tutorials, etc) to be assigned into a set of limited number of timeslots within a week, which take place in a set limited classrooms. The solution of this problem must satisfy all hard constraints without exception, whereas it may violate some soft constraints. Soft constraints are of various types, and for violation of any type a penalty value is assigned, which could be different for different types. By adding up the total value of penalties through cost function, the qualities of different solutions are measured. Several published papers on this subject have applied their approach on the benchmark problems
introduced in [9], which includes the following hard constraints:
  I. No student could be assigned to more than one course at the same time.
  II. The room should satisfy the features required by the course.
  III. The number of students attending the course should be less than or equal to the capacity of the room.
  IV. No more than one course is allowed at a timeslot in each room.

The following soft constraints, which are equally penalized, are also presented in [9]:
  I. A student has a course scheduled in the last timeslot of the day.
  II. A student has more than 2 consecutive courses.
  III. A student has a single course on a day.

The problem has
  • A set of $N$ courses, $C = \{c_1, c_2, ..., c_N\}$.
  • 45 timeslots (5 working days and 9 timeslots in every day).
  • A set of $R$ classrooms.
  • A set of $F$ classroom features.
  • A set of $M$ students.

Thus, the cost function of this problem could be considered as Eq. (1):

$$C.F = w_1 \sum_{i=1}^{N} HC_i + w_2 \sum_{j=1}^{F} SC_j,$$

where $HC_i$ is number of violation of $i^{th}$ hard constraint, $SC_j$ is number of violations of $j^{th}$ soft constraint, $w_1$ and $w_2$ are penalty values for hard and soft constraints, which are set equal to 10 and 1, respectively. This cost function simply sums up the weighted penalties of violations by the obtained solution.

III. GRAPH COLOURING HEURISTICS

Graph coloring is concerned with coloring the vertices of a given graph using a given number of colors. The relationship of graph coloring problem and timetabling is widely discussed in the literature (see [1] and [11]). Graph coloring heuristics (which are often called sequential heuristics) are used in sequential (or constructive) algorithms to order the events which are not yet scheduled, because of difficulties in scheduling them into a feasible timeslot without hard violation.

In school timetabling we say two events have a conflict if they involve the same students. The unscheduled events are represented by a graph; the events are represented by vertices and the conflicts by edges, and the difficulty of an event is determined by the number of its conflicts with other events (i.e. connected edges). The objective is to construct a timetable by scheduling the conflicting events one by one into feasible timeslots, satisfying as many of the soft constraints as possible [1]. Some of the sequential graph coloring heuristics are as follows:

- Largest degree (LD) – Events with the largest number of conflicts are scheduled first.
- Largest enrolment (LE) – Events with the largest number of student enrolment are scheduled first.
- Largest weighted degree (LWD) – Priority is given to the event with the largest weighted conflict. Each conflict is weighted based on the number of students involved in the corresponding two events.
- Random ordering (RO) – The events to be scheduled are selected randomly.
- Saturation degree (SD) – In this heuristic that is a dynamic heuristic, the next selected course to be scheduled is based on the number of available feasible timeslots. The course with the least number of available feasible timeslots will be scheduled first.

IV. MEMTIC ALGORITHMS

A. Genetic Algorithms

A GA starts by creating a random population of chromosomes, called initial population, and then these chromosomes are evaluated and sorted. A portion of chromosomes which are inferior to others are eliminated. Then two chromosomes of the remaining population are selected randomly to produce two offspring, using crossover operator, to replace the eliminated chromosomes. This reproduction (selection and crossover) continues, until the population reaches to its original size. The resultant population is called the first generation. Again the cycle of evaluation, sorting, elimination, reproduction, and mutation continues until fulfilling at least one of stopping criteria. Fig. 1 shows the flowchart of a simple GA algorithm. The stages of the GA are explained in the following.

**Initialization**: The first step to apply the GA on a problem, is encoding a problem solution as a chromosome. In the course timetabling problem, each solution should assign a timeslot and a classroom for each course. Thus, by assumption of $N$ courses, each chromosome will have a length of $2*N$ genes, of which the first $N$ genes assign the timeslot for each course and the second $N$ genes assign the

![Figure 1. Structure of GA.](image-url)
classroom for the corresponding course. Fig. 2 shows such a chromosome with 6 courses, 5 timeslots and 3 rooms. Also every generated chromosome is evaluated using Eq. (1).

Selection: Selection of parents for reproduction is carried out using the roulette wheel rank weighting method [10].

Crossover: By applying the crossover operator on two selected chromosomes, their genetic materials are swapped or combined to generate a new offspring. For this operator, we have invented the two-stage uniform crossover (2SUC), which is obtained by combining the uniform crossover and graph coloring heuristics. The illustration of Fig. 3 shows an example of the 2SUC performance.

In the first stage of the 2SUC operator, the first gene (timeslot and room) is copied from the first parent to the offspring. For copying the second gene from the second parent to the offspring, feasibility of this assignment is checked. If assignment of a gene value from any parent generates a conflict, to this gene of the offspring no value is assigned at this stage (no timeslot and no classroom). In Fig. 3 these unscheduled genes are denoted with zero values in unperfect offspring. In the second stage of applying the 2SUC operator, we use graph coloring heuristics to schedule remaining courses. First the unscheduled courses are sorted in a descending order of difficulty defined by the graph coloring heuristics. The scheduling begins from the course with the highest priority and a random feasible timeslot and room is assigned to that. For each course, if any feasible timeslot or room is not found, a random value will be assigned. The 2SUC operator generates one offspring for two selected parents.

Termination criteria: The GA continues until a given number of function evaluations are fulfilled.

B. Local Search Method

For the local search, we use a simple method similar to hill climbing. Hill climbing algorithm iteratively evaluates all of neighboring solutions and replaces the current solution by the candidate solution which results the largest increase in the solution quality. The stages of this proposed local search algorithm are shown in Fig. 4.

In this method for each chromosome \( X \), a gene is selected with a probability of \( \text{local}_\text{prob} \). A new feasible value has been assigned to selected gene and a new chromosome \( X_{\text{new}} \) is generated. If the new chromosome improves the cost of current chromosome, the current chromosome is replaced with new generated chromosome; otherwise the current chromosome is not changed. This process continues for all other genes of the current chromosome. This local search method is applied on all chromosomes for a given number of iterations.

C. Hybridizing Method

By combining the GA and local search methods, we present two memetic algorithms (MAs). In the first memetic
(MA1), the local search is applied after the mutation in the GA (see Fig. 5). Indeed the MA1 differs with the original GA only in stage 5. In the second memetic algorithm (MA2), after applying the GA for a given number of iterations (max1), the local search is applied on all members of population for a given number of iterations (max2) (see Fig. 6).

V. THE COMPUTATIONAL RESULTS

We tested the mentioned algorithms on four databases for course timetabling problem, adopted from the Metaheuristic Network website [18]. General specifications of these databases are shown in Table I, but for detail the reader is invited to visit reference [19]. In these problems it is necessary to schedule 100–400 courses into a timetable with 45 timeslots (5 work days and 9 timeslots a day), while satisfying room features and capacity constraints. These databases are divided into two groups: Small, Medium. We selected two types of Small databases and two types of Medium databases. Every algorithm was run 5 times and 100 000 function evolutions was considered as stopping criterion. The performances of algorithms were compared using two criteria:

I. The average value of solutions costs obtained in all runs (Mean).

II. The minimum value of solutions costs obtained in all runs (Min).

The considered values for different parameters of algorithms are as follows: size of initial population =60, chr_prob = 0.2, gene_prob = 0.2, local_prob = 0.4, maximum iteration number of local search in the MA1= 1, and max1 and max2 in the MA2 are considered equal to 10 and 3, respectively. The obtained results are shown in the Tables II and III.

The obtained results are shown in the Tables II and III. In the Table II, two proposed MAs are compared. Since some research work reported in the literature don't use crossover (see [12, 13]), the MA1 and MA2 algorithms shown in these tables hybridize GAs without crossover operator. Other algorithms utilize the 2SUC crossover with the heuristic written inside parenthesis.

The results in Table II show that the MA1 and MA2 algorithms which don’t use crossover, have the worst performance. Also the MA1 algorithm in all cases has a better performance than the MA2, with very few exceptions.

Among all algorithms, the MA1(SD) and MA2(SD) that use the SD heuristic inside the crossover operator has the best performance in terms of both considered criteria.

Also a comparison of the results obtained by our best algorithm i.e. the MA1(SD) with the results reported in the literature is shown in the Table III. The reported results demonstrate that on small1 the MA(SD) has the third best results, after the results of [16] and [17]. On small2 the MA1 (SD) and [16] have the second best results and finally on the Medium1 and Medium2, our proposed algorithm is the second best after [17]. The results of this table demonstrate that our proposed algorithms have a comparable performance with existing methods.

VI. CONCLUSIONS

In this paper, we proposed two types of MA for the university course timetabling problem. A new crossover method using graph coloring heuristics which is called the two-stage uniform crossover, and a local search algorithm with a mechanism similar to that of the hill climbing method were proposed. The proposed memetic algorithms were applied on some of datasets and their performance were compared with each other. The obtained results demonstrated that the MA1 has a better performance than the MA2. Also among different considered heuristics, those who
of algorithms that were used from the SD heuristic had a better performance than other algorithms. Finally a caparison among results of the best our proposed algorithm with the results reported in the literatures showed that our proposed algorithms have a comparable performance with those of already existing algorithms.

REFERENCES