Applied Soft Computing 13 (2013) 2504–2514

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Course timetabling using evolutionary operators

Danial Qaurooni [a,*], Mohammad-R. Akbarzadeh-T [b]

[a] Department of Computer Science, Amirkabir University of Technology, Tehran, Iran
[b] Center of Excellence on Soft Computing and Intelligent Information Processing, Ferdowsi University, Mashhad, Iran

ARTICLE INFO

Article history:
Received 13 September 2010
Received in revised form 21 October 2012
Accepted 24 November 2012
Available online 26 December 2012

Keywords:
Combinatorial optimization
Scheduling and timetabling
Evolutionary operators
Memetic algorithms
Grouping genetic algorithms

ABSTRACT

Timetabling is the problem of scheduling a set of events while satisfying various constraints. In this paper, we develop and study the performance of an evolutionary algorithm, designed to solve a specific variant of the timetabling problem. Our aim here is twofold: to develop a competitive algorithm, but more importantly, to investigate the applicability of evolutionary operators to timetabling. To this end, the introduced algorithm is tested using a benchmark set. Comparison with other algorithms shows that it achieves better results in some, but not all instances, signifying strong and weak points. To further the study, more comprehensive tests are performed in connection with another evolutionary algorithm that uses strictly group-based operators. Our analysis of the empirical results leads us to question single-level selection, proposing, in its place, a multi-level alternative.

1. Introduction

The essence of timetabling is scheduling a set of events, where scheduling typically consists of allotting rooms and time slots to events, subject to certain specified constraints. Varying sets of constraints constitute the gamut of timetabling subclasses that are designed to address different theoretical and real-world applications, ranging from terminal scheduling to nurse rostering. A general survey of these various subclasses is available in [1]. One such subclass, educational timetabling, deals with scheduling problems in schools and universities, with constraints pertaining to teachers, lecture hours, facilities, program conflicts, room capacities, etc. Educational timetabling itself consists of three major variants, namely the university course timetabling problem (UCTP), the exam timetabling problem (ETP), and the high school timetabling problem (HTP). These have constraint-related differences. For instance while, in the ETP, events can take place in the same room and time slot as long as they satisfy all other constraints, in the UCTP, only one event can occupy a certain room at a specified time slot. Also, while the length of the UCTP scheduling period (and hence the number of time slots) is constant, since no more than a certain number of time slots can fit in a week, ETP might be able to benefit from some level of flexibility in this area. In this paper we focus on the UCTP, but in light of the similarity in problem structure and constraint sets, the other variants may be subject to some of the issues discussed here with little loss of generality.

The constraints of the UCTP are divided into two main categories of hard and soft, denoting issues of feasibility and preference, respectively. Any violation of a hard constraint results in an infeasible timetable. Most frequent among these violations is when two events that share participants are scheduled to be held at the same time, resulting in a so-called event clash. Clearly, a timetable containing such clashes is unacceptable. Soft constraints, on the other hand, are more like considerations that, when taken into account, can yield timetables that are more accommodating for the institution, the participants, or both. Thus we might formulate a typical soft constraint to discourage the scheduling of more than two classes in a row for any student.

The NP-Completeness of the timetabling problem, has been explicitly established by Even et al. in [2], making the task of tackling the problem particularly attractive and challenging in equal measures for a host of algorithmic approaches. General surveys of the performance of algorithms can be found in [3–5]. In this paper, we focus on course timetabling problems whose solutions conform to a general evolutionary framework.

Metaheuristics have been widely utilized for solving timetabling problems. In the framework of swarm intelligence, ant colony optimization algorithms have been proposed, such as [6,7] which use ants to construct complete assignment of events to time slots using heuristics and pheromone information. Timetables are then improved using a local-search procedure, and the pheromone matrix is updated accordingly for the next iteration. Specifically, Socha et al. compare and analyze two different ant systems in [6], and go on to compare the MMAS (the better-performing ant system) with other algorithms, concluding that on the large instances, MMAS outperforms competitors. Among the more novel

* Corresponding author.
E-mail address: dani.qaurooni@aut.ac.ir (D. Qaurooni).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.11.044

approaches, honey-bee mating optimization has also been applied to both exam and course timetabling in [8]. The authors test their algorithm against Carter and Socha benchmark sets and reportedly achieve "competitive if not better" results. The popular simulated annealing (SA) heuristic has been implemented in [9–11] among others. Specifically, [10] uses a two-phased SA algorithm with a few heuristics and a new neighborhood structure that swaps between pairs of time slots, instead of two assignments. The authors report that, when tested against two standard benchmark sets, the new neighborhood heuristic improves the performance of SA. Tabu search is also applied in [12,13]. In [13], an adaptive tabu search is used to integrate an original double Kempe chains neighborhood structure, a penalty-guided perturbation operator and an adaptive search mechanism, to achieve more efficient results.

More specifically, evolutionary algorithms (EAs) have been applied to timetabling problems with varying degrees of success in the works of [14–19] among others. For example, [14] is among the first to remedy the poor performance of a genetic algorithm, compared with other conventional methods, by way of a grouping encoding. Generally, however, EAs employ other techniques to make up for their potential failure in dealing with local optima. For instance, [17] takes advantage of domain-specific heuristics in an otherwise evolutionary structure to boost performance. Working on real-world, rather than artificial, problems, Beligiannis et al. use an evolutionary algorithm to solve timetabling problems in Greek high schools in [18].

As a subclass of EAs, memetic algorithms have been used to solve timetabling in [19–24] among others. As one of the earliest applications, in [20], Burke et al. use one of two mutation operators, namely "light" and "heavy", to reschedule a random selection of events or disrupt whole time slots of events, respectively. The evolutionary process incorporates hill climbing too. Alkan and Ozcan have used a number of different one-point and uniform crossover operations, a weighted fitness function, and again a hill climbing procedure to reduce violations in [21]. A recent successful application of a memetic algorithm to laboratory timetabling is [23] where student preferences are also taken into account.

While standard benchmark sets for timetabling exist [25–27], it has been argued in [5] reasonably that "there is confusion in the field" in this regard, to the point that performing meaningful comparisons or reproducibility might be compromised. In part to overcome such obstacles, two international timetabling competitions have been held recently. The latter consisted of examination and course timetabling problem sets. McCollum et al. provide an overview of this event in [28]. Information on the winning algorithms can also be found in [29].

In general though, the standard benchmark packages are usually designed with hard *and* soft constraints in mind, hence disregarding soft constraints to focus on feasibility is hardly a challenge. In fact, many mainly focus on soft constraints, to the point that feasible solutions are found so fast as to render any comparison and analysis meaningless. In such a situation, the benchmarking instances provided in [30] address the issue of how different algorithms would fare when the focus is shifted to feasibility. This is achieved through a careful choice of a deliberate subset of a larger set of problem instances that have proven to be particularly difficult, but have at least one feasible solution. The first substantial use of this package was in relation to the work of [19], which implemented two algorithms. One of these mainly used grouping genetic operators, while the other implemented a local search. Both algorithms were augmented by local search. The average results of these two algorithms were later improved in the context of a simulated annealing algorithm [31]. In this paper, we will restrict our attention to these "hard instances", along with a discussion of the performance of the three algorithms that have made use of it.

The paper is organized in the following manner: A detailed problem description is given in the next section, followed by the outline of our memetic algorithm in Section 3, where we go over details about general algorithm structure and operator design. Section 4 includes a more focused discussion of the fitness function. In Section 5, experimental parameters are provided and the proposed algorithm is put to test. Comparisons with three other algorithms follow, demonstrating the superior performance of our memetic algorithm in two of the three instance sets. The focus is then narrowed down to evolutionary algorithms, with an analysis of operator design and development of new measures and further tests. We close by making conclusions and suggesting possible directions for future research in Section 8.

2. Problem description

Stated formally, the particular timetabling problem we study here, initially formulated for the First Timetabling Competition [32], has four parameters: T, a finite set of times; R, a finite set of rooms; E, a finite set of events; and C, a finite set of constraints, and concerns assigning times and rooms to the events so as to satisfy the required constraints. These constraints are divided into two categories of hard and soft. Generally speaking, hard constraints denote mandatory requirements that, when satisfied, produce "working" timetables. Soft constraints, on the other hand, cater to the preferences of the teachers, students, etc. Although an optimal solution attends to both hard *and* soft constraints, our algorithm ignores soft constraints and focuses on hard constraints for reasons that will be explained in Section 5.

In mathematical terms, a binary-valued function $h : S \to 0, 1$ can be associated with each hard constraint. For each solution $s \in S$, the function is defined by

$$h(s) = \begin{cases} 1 & \text{if } s \text{ does not satisfy the constraint} \\ 0 & \text{otherwise} \end{cases}$$

Let S be the set of all solutions to a given timetabling problem. A feasible solution is any solution $s \in S$ that satisfies all hard constraints. Thus, the objective function might be formulated as

$$F(S) = \sum_{i=1}^{n} h_i(S) \tag{1}$$

where hard constraints are given by $h_1, h_2, \ldots, h_n$. Our particular formulation includes the following hard constraints:

- H1: no student is permitted to attend more than one event at any one time;
- H2: only one event is scheduled for any room and any time slot;
- H3: all of the features required by the event should be satisfied by the room, which has an adequate capacity.

Aside from general problem description, it might be useful to discuss a problem-solving perspective that will influence our solution-building later on, namely the idea of interpreting timetabling as a grouping problem. In [33], Falkenauer defines grouping problems as those where the task is to partition a set of objects U into a collection of mutually disjoint subsets u_i of U, where

$$\bigcup u_i = U \quad \text{and} \quad u_i \cap u_j = 0, i \neq j \tag{2}$$

according to a set of problem-specific constraints that define allowable groupings. Familiar cases where this criterion holds include bin packing, graph coloring and timetabling. A corresponding grouping genetic algorithm (GGA), devised by Falkenauer is based on the notion that, while traditional genetic operators are well-suited for