Computers & Operations Research 52 (2014) 203–208

Contents lists available at ScienceDirect

Computers & Operations Research

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

Variable Neighborhood Search based algorithms for high school timetabling

George H.G. Fonseca [a], Haroldo G. Santos [b]

[a] Computing and Systems Department, Federal University of Ouro Preto, Brazil
[b] Computing Department, Federal University of Ouro Preto, Brazil

ARTICLE INFO

Available online 6 February 2014

Keywords:
Variable Neighborhood Search
High School Timetabling Problem
Third International Timetabling Competition

ABSTRACT

This work presents the application of Variable Neighborhood Search (VNS) based algorithms to the High School Timetabling Problem. The addressed model of the problem was proposed by the Third International Timetabling Competition (ITC 2011), which released many instances from educational institutions around the world and attracted 17 competitors. Some of the VNS algorithm variants were able to outperform the winner of Third ITC solver, which proposed a Simulated Annealing – Iterated local Search approach. This result coupled with another reports in the literature points that VNS based algorithms are a practical solution method for providing high quality solutions for some hard timetabling problems. Moreover they are easy to implement with few parameters to adjust.

1. Introduction

The High School Timetabling Problem is faced by many educational institutions around the world. The basic search version consists in assigning teacher × class activities to timeslots and rooms in such a way that no teacher, class or room is involved with more than one event at time. Generally, this assignment is repeated weekly until the end of the semester. Many other constraints are considered in real problems, like availability of teachers, to avoid idle times and to limit the number of lessons of the same subject taught to a class in a day.

Beyond its practical importance, this problem was proven to be $\mathcal{NP}$–Hard [1,2]. Progress in heuristic and exact approaches for tackling these problems is a major goal of current research in Operations Research and Artificial Intelligence.

Three international competitions (ITCs) were made to bring the attention of scientists and practitioners for this problem, with the objective of performing comparisons of different methods in a controlled computational environment: the first one happened in 2003 [3] and was won by Kostuch [4] with a 3-phase Simulated Annealing (SA) [5] based approach. In 2007, the second one [6] started and was composed of three separated tracks, which were mostly won by Müller [7] also with a Simulated Annealing based approach. The last one [8] happened in 2012 and was won by a Simulated Annealing – Iterated Local Search [9] approach.

As the results of the competitions show, local search methods are defining the state-of-art heuristic solvers for educational timetabling problems. Specially, the Simulated Annealing metaheuristic composed the solver of all winners. The role of exact methods which employ Integer Programming, such as the proposed in [10–12], appears to be still very limited for tackling the problems and instances which appeared in these competitions, considering the absence of these techniques in submissions. This scenario contrasts with the first International Nurse Rostering Competition [13], for instance, where two of the first places used Integer Programming in some form.

This paper presents a computational study of Variable Neighborhood Search and its variants applied to the Third ITC problem. The results indicate that the proposed method outperforms the state-of-art method.

The remaining of this work is organized as follows: Section 2 presents the problem considered in this paper, the Third ITC problem; Section 3 presents our solution approach; Section 4 presents computational experiments and finally, Section 5 concludes our paper and discusses future works.

2. High School Timetabling Problem model

The roots of the School Timetabling model considered in this paper, the model of the Third ITC, are in the Benchmarking project for (High) School Timetabling.[1] The project, which involved a group of researchers in this area, started with the ambitious goal of developing a XML format capable of modeling different school

E-mail addresses: george@decsi.ufop.br (G.H.G. Fonseca), haroldo@iceb.ufop.br (H.G. Santos).

http://dx.doi.org/10.1016/j.cor.2013.11.012
0305-0548 © 2014 Published by Elsevier Ltd.

[1] http://www.utwente.nl/ctit/hstt/

timetabling problems arising in diverse institutions around the world. Initial versions of this project appeared in the PATAT 2008 conference [14], with an improved version named XHSTT published later [15]. Nowadays, the project site holds approximately 50 datasets from 11 countries. The project site also includes an evaluator to validate solutions and the best known solutions are kept, so that the results of newly proposed methods can be immediately confronted with previously obtained results. Some of the previous models which are now in XHSTT are [16–20,10,21]. The model is split into three main entities: Time and Resources, Events and Constraints. A solution consists of a set of assignments of times and resources to the events.

2.1. Times and resources

The time entity consists of a timeslot, which is an indivisible interval of time. Timeslots do not overlap and can be grouped in timegroups. Resources are entities which attend events. Typical resources are students, teachers and rooms [21]:

Students: a group of students attends events (lessons); important constraints associated with students are the control of their idle times and the number of lessons taken by day.
Teachers: teachers perform their academic tasks in events; the allocation of teachers for specific teaching activities can be preassigned or not; when teachers are not preassigned, they should be assigned according to their qualifications and workload limits.
Rooms: the usage of rooms for hosting events must be observed: some events require rooms with a given capacity and/or a set of special features.

2.2. Events

An event (*instance event*) is a meeting between resources, usually representing a simple *lesson* or a set of lessons (event group). Each instance event needs to be scheduled into one or more *solution events*. Timeslot assignments to events are called *meets* and the assignment of resources to events is *tasks*. The term *course* is used to designate a group of students who attend to the same events. Other kinds of events, like meetings, are allowed by the model [21]. The following attributes can be specified for events, the first one is the only obligatory:

Duration: represents the number of timeslots which have to be assigned to the event.
Course: a course is a grouping of events: events declared in the same course constitute a course of study in one subject for one group of students.
Pre-assigned resources: to attend the event.
Workload: that will be added to the total workload of resources assigned to the event.
Pre-assigned timeslot: some events have only one timeslot in which they can be assigned.

2.3. Constraints

Post et al. [21] group the constraints into three categories: basic constraints of scheduling, constraints of events and constraints of resources. The objective function $f(\cdot)$ is computed considering the summation of penalties for deviations in different constraints and events/resources which they refer. The flexibility of XHSTT allows the inclusion of non-linear terms in the cost function which is used to compute the penalties [15]. The constraints are also divided into hard constraints, whose satisfaction is mandatory; and soft constraints, whose satisfaction is desirable but not obligatory. Costs for violations in these two types of constraints are summed in two separated costs: the infeasibility cost and the quality cost, defining a hierarchical objective function. Each instance can define whether a constraint is hard or soft, its weight and the type of cost function used (eg. linear or quadratic). For more details, see [15].

2.3.1. Basic constraints of scheduling

1. Assign Time: assign timeslots to each event.
2. Assign Resource: assign the resources to each event.
3. Prefer Times: indicates that some event have preference for a particular timeslot(s).
4. Prefer Resources: indicates that some event have preference for a particular resource(s).

2.3.2. Constraints to events

1. Link Events: to schedule a set of events to the same starting time.
2. Spread Events: specify the allowed number of occurrences for event groups in time groups between a minimum a maximum number of times; this constraint can be used, for example, to define a daily limit of lessons.
3. Avoid Split Assignments: for each event, assign a particular resource to all of its meets.
4. Distribute Split Events: for each event, assign between a minimum and a maximum meets of a given duration.
5. Split Events: limits the number of non-consecutive meets that an event should be scheduled and its duration.

2.3.3. Constraints to resources

1. Avoid Clashes: assign the resources without clashes (i.e. without assign the same resource to more than one event at a timeslot).
2. Avoid Unavailable Times: avoid assigning resources on the times that they are not available.
3. Limit Workload Constraint: schedule the workload of the resources between a minimum and a maximum bound.
4. Limit Idle Times: the number of idle times in each time group should lie between a minimum and a maximum bound for each resource; typically, a time group consists of all timeslots of a given week day.
5. Limit Busy Times: the number of busy times in each time group should lie between a minimum and a maximum bound for each resource.
6. Cluster Busy Times: the number of time groups with a timeslot assigned to a resource should lie between a minimum and a maximum limit; this can be used, for example, to concentrate teacher's activities in as few days as possible.

3. Solution approach

Our approach uses the Kingston High School Timetabling Engine (KHE) [22] to generate initial solutions. Afterwards, we implemented the Variable Neighborhood Search metaheuristic and some of its variants to perform local search around this solution. These elements will be explained in the following subsections.

3.1. Build method

The KHE is a platform for handling instances of the addressed problem. It also provides a solver, used to build initial solutions in