Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Discrete Mathematics 39 (2012) 169–176

Electronic Notes in DISCRETE MATHEMATICS

www.elsevier.com/locate/endm

A SA-VNS approach for the High School Timetabling Problem

Samuel S. Brito [1] George H. G. Fonseca [2] Tulio A. M. Toffolo [3]
Haroldo G. Santos [4] Marcone J. F. Souza [5]

Computing Department, Federal University of Ouro Preto, Ouro Preto, Brazil

Abstract

The High School Timetabling Problem consists in assigning timeslots and resources to events, satisfying constraints which heavily depend on the specific institution. This work deals with the problem of the ongoing III International Timetabling Competition (ITC), which includes a diverse set of instances from many educational institutions around the world. We proposed an approach based on Simulated Annealing and Variable Neighborhood Search metaheuristics. One important structural feature of our approach is the use of the Kingston's High School Timetabling Engine (KHE) to generate initial solutions combined with the multi-neighborhood search. Such approach led us to the finals of the ongoing competition.

Keywords: Variable Neighbourhood Search, Simulated Annealing, High School Timetabling Problem, Third International Timetabling Competition

[1] Email: samuelsouza@iceb.ufop.br
[2] Email: george@decea.ufop.br
[3] Email: tulio@toffolo.com.br
[4] Email: haroldo@iceb.ufop.br
[5] Email: marcone.freitas@gmail.com

1571-0653/$ – see front matter
doi:10.1016/j.endm.2012.10.023

1 Introduction

The High School Timetabling Problem, denoted as Class×Teacher Timetabling Problem in the early works of Gotlieb [3], consists in the production of a schedule in such a way that no teacher or class attend more than one lesson at same time, as well as respecting other constraints.

The automated school timetabling has been the subject of much research in the fields of Artificial Intelligence and Operational Research. Moreover, the problem addressed is classified as $\mathcal{NP}$-Hard [2]. Methods based in Integer Programming were proposed to the problem [11], but they can only solve a small subset of instances of the problem in feasible processing time. Nowadays, metaheuristic methods are commonly applied to the problem, like Simulated Annealing [9], Iterated Local Search [1] and Tabu Search [8].

In this sense, the main goal of this work is to present a study of an algorithm that combines both the Simulated Annealing and the Variable Neighbourhood Search metaheuristics, applied to the High School Timetabling Problem proposed by the ongoing Third International Timetabling Competition 2011 (ITC 2011).

2 High School Timetabling Problem

Used in the ITC 2011, the addressed model of High School Timetabling Problem came up with the goal of providing a generic model capable to address the various features of the High School Timetabling Problem around the world [12] [11] [10].

The model is split into four main entities: times, resources, events and constraints. The time entity consists of a timeslot or a set of timeslots (time group). The resources, in turn, are divided in three main categories: students, teachers and rooms [10]. An event is the basic unit of allocation, representing a simple lesson or a set of lessons (event group). A timeslot assignment to a event is called meet and a resource assignment to a event is called task. Other kinds of events, like meetings, are allowed by the model [10].

2.1 Constraints

Post et al. [10] group the constraints in three categories: basic constraints of scheduling, constraints to the events and constraints to the resources. The objective function $f(.)$ is calculated in terms of violations to each constraint penalized according to their weight (so this is a minimization problem). They

were also divided in hard constraints, whose attendance is mandatory, and soft constraint, whose attendance is desirable but not mandatory. Each instance defines whether a constraint is considered to be hard or soft. For more details, see [10].

3 Solution Approach

Our approach uses the Kingston's High School Timetabling Engine (KHE)[6] to generate initial solutions. In sequence, Simulated Annealing metaheuristic is used to improve the initial solution. Finally, Variable Neighborhood Search is applied to perform local search around the solution obtained by Simulated Annealing. 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 which also provides a solver, used to build initial solutions in the presented approach. The KHE's solver was chosen to generate the initial solutions since it is able to find a solution in a very small amount of time.

The incorporated solver is based on the concept of Hierarchical Timetabling [5], where smaller allocations are joint to generate bigger blocks of allocation until a full representation of the solution is developed. For full details, see [5,6].

3.2 Neighborhood Structure

Six neighborhood structures were used in our local search procedures:

(i) Event Swap (ES): two events e_1 and e_2 have their timeslots t_1 and t_2 swapped;

(ii) Event Move (EM): an event e_1 is moved from t_1 to another timeslot t_2;

(iii) Event Block Move (EBM): like es, swaps the timeslot of two events e_1 and e_2, but if the events have different duration, e_1 is moved to the following the last timeslot occupied by e_2.

(iv) Kempe Move (KM): two times t_1 and t_2 are fixed and one seeks the best path at the bipartite conflict graph containing all events in t_1 and t_2; arcs are build from conflicting events which are in different timeslots and their cost is the cost of swapping the timeslots of these two events.

(v) Resource Swap (RS): two events e_1 and e_2 have their assigned resources

r_1 and r_2 swapped. Resources r_1 and r_2 should play the same role (e.g. both have to be teachers).

(vi) Resource Move (RM): an event e_1 have his assigned resource r_1 replaced to a new resource r_2.

3.3 Simulated Annealing

Proposed by Kirkpatrick et al. in [7], the metaheuristic Simulated Annealing is a probabilistic method on an analogy to thermodynamics simulating the cooling of a set of heated atoms. This technique starts its search from any initial solution. The main procedure consists of a loop that randomly generates, at each iteration, one neighbor $s^{'}$ of the current solution s.

The developed implementation of Simulated Annealing is described in Algorithm 1. The considered parameters were $alpha = 0.97$ and $T_0 = 5$. The parameter SA_{max} was defined according to the number of events (nE) for each instance set by a multiplier. If the initial solution is feasible (i.e. there is no hard constraint violation), $SA_{max} = nE \times 10$, otherwise, $SA_{max} = nE \times 100$.

Algorithm 1: *Simulated Annealing*
Input: $f(.)$, $N(.)$, α, SA_{max}, T_0, s, *timeout*.
Output: Best solution s *found.*
$s^* \leftarrow s$;
$IterT \leftarrow 0$;
$T \leftarrow T_0$;
while $T > 0$ *and* $elapsedTime < timeout$ ***do***
  while $IterT < SA_{max}$ ***do***
    $IterT \leftarrow IterT + 1$;
    Generate a random neighbor $s^{'} \in N(s)$;
    $\Delta = f(s^{'}) - f(s)$;
    if $\Delta < 0$ ***then***
      $s \leftarrow s^{'}$;
      if $f(s^{'}) < f(s^*)$ ***then*** $s^* \leftarrow s^{'}$;
    else
      Take $x \in [0, 1]$;
      if $x < e^{-\Delta/T}$ ***then*** $s \leftarrow s^{'}$;
  $T \leftarrow \alpha \times T$;
  $IterT \leftarrow 0$;
$s \leftarrow s^*$;
return s;

At each iteration the selected movement can be from any of the proposed neighborhoods. If the instance requires the resource assignment, the neighborhood is chosen based on the following probabilities: $es = 0.2$, $em = 0.4$, $ebs = 0.1$, $rs = 0.2$ and $rm = 0.1$, otherwise, the neighborhoods rs and rm are not used and the odds are: $es = 0.5$, $em = 0.3$ and $ebs = 0.2$. These values were empirically adjusted.

3.4 Variable Neighbourhood Search

Proposed by Hansen et al. in [4], the Variable Neighborhood Search is a local search method which consists in exploring the search space by systematics changes on the neighborhood structure. It allows a change of the neighborhood structures within this search.

The search stops when the timeout is reached, returning the best solution found. The VNS implementation is described in Algorithm 2, where $f(.)$ denotes the objective function, s denotes the initial solution obtained, k_{max} represents the number of neighborhood structures and t_{max} the maximum execution time. Variable $N_k(s)$ represents the set of solutions in the kth neighborhood of s. The function $Shake$ generates a random neighbor $s^{'}$ in the kth neighborhood of s, i.e., $s^{'} \in N_k(s)$ [4].

Algorithm 2: *Variable Neighborhood Search*
Input: $f(.)$, s, k_{max}, t_{max}.
Output: Best solution s found.
repeat
  $k \leftarrow 1$;
  repeat
    $s^{'} \leftarrow Shake(s, k)$;
    $s^{''} \leftarrow LocalSearch(s^{'})$;
    if $f(s^{''}) < f(s)$ ***then***
      $s \leftarrow s^{''}; k \leftarrow 1$;
    else $k \leftarrow k + 1$;
  until $k = k_{max}$;
  $t \leftarrow CpuTime()$;
until $t > t_{max}$;
return s;

In our implementation we change the neighborhood structure following the order defined at Section 3.2. These neighborhoods are organized according to their computational complexities.

Furthermore, we use two variants of this method: RVNS (Reduced Variable Neighborhood Search) and GVNS (General Variable Neighborhood Search). The RVNS method is obtained if random points are selected according to the current neighborhood and no local search is made. Rather, the values of these new solutions are compared with that of the incumbent solution, which is updated in case of improvement. The GVNS method is obtained if the Local Search step is replaced by a Variable Neighborhood Descent (VND) method. The Variable Neighborhood Descent method performs the change of neighborhoods in a deterministic way. In this algorithm, the final solution should be a local minima with respect to all $k^{'}_{max}$ neighborhoods.

4 Computational Experiments

All experiments ran on an Intel ® Core i7 3.07 Ghz computer with 8GB of RAM memory running Linux openSUSE 12.1 64-bits. The programming language used on software development was C++ compiled by GCC 4.6.1. The processing time was adjusted according to the benchmark available from Third International Timetabling Competition 2011, which in our case was 680 seconds (normalized). All of our results was validated by HSEval validator [6].

The set of instances available from ITC 2011 [7] was originated from many countries and ranges from small instances to huge challenging ones.

4.1 Obtained Results

To compare the performance of the VNS method, we used the same solutions provided initially by KHE and refined by Simulated Annealing. The Table 1 presents the obtained results by GVNS and RVNS metaheuristics. The results were expresed by the pair x/y, where x represents the infeasibility measure of a solution and y represents the quality measure. Column *Duration* refers to the total duration of all events in the dataset. The column *KHE* contains the initials solutions provided by KHE engine and the column *SA* contains the improved solution obtained after running the Simulated Annealing. The last two columns contains the results obtained by the execution of GVNS and RVNS respectively.

4.2 Discussion of Results

Our heuristic approach used the KHE solver to build an initial solution and two metaheuristics to refine it. The KHE solver was able to quickly find initial solutions and the Simulated Annealing improved them. The proposed neighborhood structures were able to consistently explore the solutions space and perform significant improvements after both the KHE and Simulated Annealing were applied.

From small to medium instances, the GVNS approach returned better results than the RVNS one. It was able to significantly reduce the soft cost of most instances. For larger instances the RVNS method reached better or equivalent solutions to those obtained by the GVNS. This shows that the VND method becomes expensive as the size of the instances grow.

[6] http://sydney.edu.au/engineering/it/~jeff/hseval.cgi
[7] http://www.utwente.nl/ctit/hstt/archives/XHSTT-2012

Instance	Duration	KHE	SA	GVNS	RVNS
BGHS98	1564	11 / 476	11 / 475	11 / 475	11 / 475
SAHS96	1876	22 / 31	18 / 72	18 / 52	18 / 52
TES99	806	9 / 187	9 / 187	9 / 187	9 / 187
BrazilInstance1	75	0 / 81	0 / 50	**0 / 21**	0 / 44
BrazilInstance4	300	17 / 225	12 / 222	**12 / 123**	12 / 153
BrazilInstance5	325	13 / 249	4 / 295	**4 / 148**	4 / 184
BrazilInstance6	350	11 / 339	4 / 327	4 / 213	4 / 213
BrazilInstance7	500	24 / 287	11 / 489	**11 / 267**	11 / 318
StPaul	1227	2 / 49068	2 / 49068	2 / 48758	**2 / 48450**
FinlandArtificialSchool	200	20 / 14	19 / 12	19 / 12	19 / 12
FinlandCollege	854	21 / 749	1 / 838	**1 / 49**	1 / 77
FinlandHighSchool	297	7 / 446	0 / 189	**0 / 16**	0 / 73
FinlandSecondarySchool	306	38 / 353	0 / 400	**0 / 114**	0 / 129
GreecePatras3rdHS2010	340	14 / 367	0 / 149	**0 / 12**	0 / 20
GreecePreveza3rdHS2008	340	13 / 603	0 / 198	0 / 37	**0 / 33**
ItalyInstance1	133	0 / 323	0 / 279	**0 / 20**	0 / 31
Kottenpark2003	1203	1 / 72413	0 / 88387	1 / 72413	**0 / 85372**
Kottenpark2005	1272	24 / 23206	20 / 28482	20 / 28710	**20 / 28482**
Lewitt2009	838	365 / 0	0 / 144	0 / 78	**0 / 74**

Table 1
Obtained Results

For some instances, even the production of feasible solutions is a very hard task. These instances commonly define most of constraints as hard ones. Therefore, ITC 2011 do not expect that a solver find all feasible solutions and so the use of the pair $infeasibility/quality$ was recommended.

5 Concluding Remarks

According to the results presented in Section 4, we can conclude that GVNS method generates good solutions for smaller instances, while the RVNS method is the most useful for larger instances, as previously reported by [4]. Our approach to solve high school timetabling Problem reached good results: we obtained feasible solutions for eight of the nineteen instances tested. It is important to note that the Timetabling International Competition 2011 is still ongoing, so we did not have access to other solutions to compare our results. But since we are one of the finalists of the competition, we may conclude that

our approach is competitive.

References

[1] Barbosa, S. H., and S. R. Souza, *Resolução do problema de programação de cursos universitários baseada em currículos via uma meta-heurística híbrida grasp-ils-relaxado*, XLIII Simpósio Brasileiro de Pesquisa Operacional, Proceedings of XLIII SBPO **43** (2011), 2827–2882.

[2] Garey, M. R., and D. S. Jonhson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, San Francisco, CA, USA, 1979.

[3] Gotlieb, C. C., *The construction of class-teacher time-tables*, Proc. IFIP Congress 62 (1963), 73–77.

[4] Hansen, P., N. Mladenović, and J. A. M. Pérez, *Variable neighborhood search: methods and applications*, 4OR: A quarterly journal of operations research **6** (2008), 319–360.

[5] Kingston, J. H, *Hierarchical timetable construction*, Proceedings of The 6th International Conference on the Practice and Theory of Automated Timetabling (2006), 196–208.

[6] Kingston, J. H., "A software library for school timetabling", URL: http://sydney.edu.au/engineering/it/ jeff/khe/.

[7] Kirkpatrick, S., D. C. Gellat, and M. P. Vecchi, *Otimization by simulated annealing*, Science **220** (2003), 671–680.

[8] Lú, Z., and J. K. Hao, *Adaptive tabu search for course timetabling*, European Journal of Operational Research **200** (2010), 235–244.

[9] Muller, T., *Itc2007 solver description: a hybrid approach*, Annals of Operations Research **172** (2009), 429–446.

[10] Post, G., S. Ahmadi, S. Daskalaki, J. H. Kingston, J. Kyngas, C. Nurmi, and D. Ranson, *An xml format for benchmarks in high school timetabling*, Annals of Operations Research **194** (2010), 385–397.

[11] Santos, H. G., E. Uchoa, L. S. Ochi, and N. Maculan, *Strong bounds with cut and column generation for class-teacher timetabling*, Annals of Operations Research **194** (2012), 399–412.

[12] Wright, M., *School timetabling using heuristic search*, Journal of Operational Research Society **47** (1996), 347–357.