Combining the Min-Conflicts and Look-Forward Heuristics to Effectively Solve A Set of Hard University Timetabling Problems

Vincent Tam and David Ting

Department of Electrical and Electronic Engineering
University of Hong Kong, Pokfulam, Hong Kong.
vtam@eee.hku.hk

Subject areas : University Timetabling, Local Search Methods, Scheduling.

Abstract

University timetabling problems (UTPs) represent a class of challenging, high-dimensional and multi-objectives combinatorial optimization problems that are commonly solved by constructive search, local search methods or their hybrids. In this paper, we proposed to combine the min-conflicts and look-forward heuristics used in local search methods to effectively solve general university timetabling problems. Our combined heuristics when augmented with the k-reset operator, and appropriate heuristic variable ordering strategy achieved impressive results on a set of challenging UTPs obtained from an international timetabling competition. A preliminary analysis of the results was given. More importantly, our search proposal shed light on effectively solving other complex or large-scale scheduling problems.

1 Introduction

Timetabling problems represents a general class of challenging, high-dimensional and multi-objectives combinatorial optimization problems that are NP-complete [7]. School or university timetabling problems (UTPs) with fixed numbers of timeslots can be formulated as discrete constrained optimization problems (COPs) [4, 7] with their systematic framework to capture users' requirements as constraints to be handled.

In general, there are basically two approaches to solve COPs or specfically UTPs. The first is the traditional constructive search approach that systematically extends the current variable assignments, often based on a search tree, until a complete and consistent solution is returned. An example is the branch-and-bound (BnB) method [7]. The second approach involves some kinds of local search methods such as the GA or SA working on an initial and complete solution with a number of iterative repairs until a predetermined resource like the maximum number of iterations is exhausted. Some researchers combined both approachces as search hybrids [7]. Nevertheless, many existing local search methods or their hybrids are based on the min-conflicts heuristics proposed by Minton et. al [4] for solving constrained problems. The min-conflicts heuristic (MCH) is to examine each variable to assign a value with the minimum number of constraint violations. MCH had remarkable success in solving many challenging COPs or constraint satisfaction problems (CSPs) including the car-sequencing [7] or n-queens problem [4]. A previous work by Yoshikawa et. al [8] only focused on using the MCH to generate an initial solution for solving both school and university timetabling problems. After a fairly good-quality initial solution is generated, their proposal relies on a heuristic billiard-move operator to iteratively repair on the current and complete assignment of lessons for school/university timetabling. Later, Kwan et. al [2] have attempted various lesson or timeslot selection heuristics to try to improve the performance of the billiard-move based heuristic search methods in handling real-life instances of secondary school timetabling problems (STPs) in Hong Kong.

After all, no previous work has ever considered to *actively* apply the MCH as a repair heuristic during the search process for solving school/university timetabling problems. In pursuit of this interesting idea, we followed a previous MCH-based work proposed by Stuckey and Tam [5] to combine the MCH and look-forward heuristics to test on a set of hard graph-coloring problems [1] since timetabling problems can be formulated as graph-coloring problems or their variants. With some preliminary success on graph-coloring [6], we quickly adapted our combined heuristic search framework to successfully solve a set of 20 challenging UTPs obtained from the PATAT international timetabling competition [3]. Undoubtedly, our proposal of combined heuristics not only gained remarkable success in effectively solving ALL 20 instances of challenging UTPs but also shed light on solving other complex or large-scale scheduling problems.

This paper is organized as follows. Section 2 clearly defines the challenging UTPs that we are interested in. In Section 3, we describe our search proposal of effectively combining the MCH and look-forward heuristics to tackle the challenging UTPs or possibly other school timetabling problems as well. Section 4 gives the empirical evaluation of our search proposal on the set of challenging UTPs. Lastly, we conclude our work in Section 5.

2 University Timetabling Problems

The 20 challenging university timetabling problems (UTPs) were obtained from the International Timetabling Competition 2002 [3]. The competition was organized by the Metaheuristics Network and sponsored by the International Series of Conferences on the Practice and Theory of Automated Timetabling (PATAT) from October 2002 to March 2003.

A major reason that we were interested in the 20 challenging UTPs was simply because ALL 20 instances in the competition were reductions of various typical university timetabling problems. All challenging instances of UTPs adopted a 5-day cycle with 9 periods per day. In each problem instance, the total number of lessons, each with its corresponding list of students, a list of required features for each lesson, the total number of available rooms, the corresponding room capacities, and finally the corresponding lists of room features were clearly specified. Besides, there were two basic categories of constraints: *hard* or *soft*. The hard constraints included:

- Any student should not attend more than one lesson at any time.
- There should be at most one lesson scheduled at each room at any time.
- The room assigned to a lesson should contain all the features required by the lesson.
- The room should be large enough for the lesson assigned.

The soft constraints were:

- Any student should not have a lesson at the last period of a day.
- Any student should not have more than two consecutive periods at any time in a day.
- Any student should not attend only one lesson in a day.

There is at least one perfect solution for each of the 20 challenging UTPs. Therefore, the designated algorithm should be able to find solutons for all 20 instances without violating any hard constraints within a predetermined time limit depending on a specific combination of the hardware and operating systems used.

3 Combining the Min-Conflicts and Look-Forward Heuristics

Since most existing work is too specific in design or inappropriate for student-level reasoning, we devise a new combination of the min-conflict [4] and look-forward [5] heuristics as a general-purpose search scheme for tackling the challenging UTPs. Unlike Yoshikawa's proposal [8] that relied on the full look-ahead (arc-consistency) technique to produce a good-quality initial solution to be iteratively improved by the min-conflicts hill-climbing (MCHC) heuristic only, our search proposal starts with a randomized initial solution and aggressively uses both MCHC and the more intelligent look-forward heuristics. The MCHC is used to bias toward any local minimum of constraint violations while the look-forward, as originally proposed by Stuckey and Tam [5], aims to guide any strongly biased heuristic search more intelligently by trying different plausible neighboring states to sensibly break ties during the search. Together, the MCHC and look-forward heuristics complement each other with impressive results achieved on a set of hard graph-coloring problems [6]. In this work, we quickly adapt the original look-forward heuristic operator through an integration with the interesting billiard-move operator as suggested in Yoshikawa's work for handling UTPs. Besides, we try out several heuristic variable ordering techniques and find that the minimal width ordering (MWO) [7] heuristic strategy, giving all variables a total ordering with the minimal width, is most effective in guiding our proposed search scheme to solve the challenging UTPs.

Our proposed search algorithm combining the min-conflicts and look-forward heuristics to effectively handle general UTPs can be divided into 3 major parts. The first part is mainly used for initializing the relevant data structures, domains and variables; preparing of the heuristic minimal width ordering (MWO) [7] of all variables; and properly setting up a queue of variables with constraint violations and a tabu-list to avoid cycling before the search starts. The second part represents the main body of the search algorithm in which we firstly apply the MCHC followed by the adapted look_forward_billiard_movement operator when no "recent" improvement on the previous solution is made. The last part denotes the constraint relaxation codes which provides flexibility to relax the more difficult soft constraints in order to focus the search effort on the more important constraints and finally return the best satisfactory solution ever found. Clearly, on certain easy UTPs, the last part of our search algorithm can be flexibly removed. The major operators of our search proposal are explained below. For detail, refer to [6].

- The apply_MCH operator: a relatively expensive operator in terms of both time and space complexities. It performs the steepest-descent step by examining all the all the values in the domain of a variable. The value that causes the minimum number of constraint violations will be selected. Ties are broken randomly.

- The look_forward_billiard_movement operator: the original look_forward operator is proposed to carefully examine the ties resulting from the apply_MCH operator. Basically, the look_forward operator works by finding a value from the ties to intelligently guide the search towards a more promising solution. When ties are formed, the concerned variable will be assigned to every single value in the ties with the remaining variables being modified by a low-cost heuristic operator. The new assignment causing the smallest number of constraint violations will lastly be selected. Since Yoshikawa et. al [8] have already proposed the billiard_ movement operator to pull the landscape away from local minima in timetabling problems, it is sensible to use the billiard_operator as a low-cost heuristic operator to be integrated into our original look_forward method as the new look_forward_billiard_movement operator.

4 Experimental Results

In this section, we will summarize the performance of our search proposal combining both min-conflicts and look-forward heuristics on a challenging set of 20 university timetabling problems (UTPs) [3]. Our prototype was implemented in Java and executed on an Intel Pentium 700Mhz machine under the Microsoft Windows XP platform. It is interesting to note that the competition organizers prepared a program to determine the time allowed for solving each problem instance depending on the combination of machine and operating system used. With respect to our specific settings, the allowed time was set to be $1,753$ CPU seconds.

Table 1 gives the results of our search proposal on all instances of the challenging UTPs. It should be noted that a "*" symbol in the last column showed that a subset of the end-of-day constraints was relaxed while those cases marked with "**" indicated that both consecutive-lesson and single-lesson constraints were relaxed. Any non-zero penalty implies that an overall solution satisfying all the hard and soft constraints imposed cannot be found. Basically, feasible solutions without violating any hard constraint were successfully found by our effective search proposal on ALL 20 challenging UTPs within the allowed time set by the timetabling competition organizers, ranging from the minimum CPU time of 9.92 seconds for the competition07.tim case to the maximum 1600 CPU seconds for the competition10.tim with an average performance of 385.33 CPU seconds over all test cases. Clearly, the total time required to solve each case depends on the constrain-ness of both hard and soft constraints, all features of the resources (including students and rooms) involved, and the overall size of the search space.

Prob.	CPU sec.	Best Penalty	Remarks
comp01.tim	683.77	24	*
comp02.tim	655.18	27	*
comp03.tim	505.67	30	*
comp04.tim	186.97	0	**
comp05.tim	78.40	0	**
comp06.tim	45.70	0	**
comp07.tim	9.92	0	**
comp08.tim	306.65	18	*
comp09.tim	440.13	34	*
comp10.tim	1600.00	3	**
comp11.tim	753.23	30	*
comp12.tim	469.26	0	**
comp13.tim	370.50	0	**
comp14.tim	37.81	0	**
comp15.tim	68.43	0	**
comp16.tim	267.31	24	*
comp17.tim	163.29	0	**
comp18.tim	290.01	32	*
comp19.tim	352.54	0	**
comp20.tim	421.88	20	*

Table 1: Our Obtained Results on the 20 UTPs.

The apply_MCH operator performed generally well on the challenging UTPs. The major weakness of our proposed search algorithm was the lack of suitable repair operators when the whole search landscape was trapped into local minima. The look_forward_billiard_movement, though successfully improve the solution quality to a large extent, still sometimes failed to further improve the search landscape when the corresponding penalty values dropped below a specific value as shown by the plotting of the variations of penalty values against the number of iterations used by our proposed method. When examining the competition04.tim case, our search algorithm actually goes through a very rugged landscape to drastically decrease the penalty value from the initial 120 to around 10 in the first 2000 iterations, and successfully finds a feasible solution to all hard and relevant soft constraints after another 5000+ iterations. On the other hand, for the competition10.tim case, our search proposal exhibits a very sharp drop in penalties in the first 3000 iterations, later followed by another slow drop around $14,000$ to $18,000$ iterations, and lastly remains almost level off from $25,000$ iterations onwards. The specific reason(s) behind this interesting and opposite phenomenon prompt us for further investigation. Furthermore, we have investigated two possible improvements of the look_forward_billiard_movement operator to perform double or triple billiard movement. None of them shows any significant improvement, and thus accords to the previous empirical experience as reported by Yoshikawa et. al [8]. Furthermore, we experiment with a variant of our search proposal using Yoshikawa's RFLG algorithm to generate an initial solution. Again, the variant fails to show any improvement.

5 Conclusion

Yoshikawa et. al [8] proposed a combined search method using the *Really-Full-Lookahead-Greedy* (RFLG) algorithm to generate a good-quality initial solution, and the strongly biased Min-Conflicts Hill-Climbing (MCHC) [4] to iteratively repair the current solution until a satisfactory solution was produced. In this paper, instead of relying on the RFLG or other initialization method to produce a good-quality initial solution, we proposed to combine both MCHC and the intelligent look-forward to *aggressively* guide the search for *better* improvements from the current search position until a feasible and (sub-)optimal solution is obtained, or any resource exhausted.

We have tested prototypes of our proposed search algorithm and its variants running on a Pentium 700Mhz machine installed with the Windows XP operating system to solve a set of 20 hard UTPs obtained from the PATAT International Timetabling Competition. The prototype of our search proposal combining both MCHC and look-forward heuristics successfully solve ALL 20 UTPs within the time limit set by the organizers. Our research team was short-listed as 1 of the 21 teams that could successfully solve ALL instances of the hard UTPs all over the world. More importantly, we conducted a preliminary analysis on the performance of our search proposal on these 20 UTPs. Obviously, there is still much room to further improve our search proposal.

First, an investigation of a more suitable performance indicator, possibly applying the concept of "utility" to estimate the relative gain, to effectively minimizing "more promising" features before their relaxation is interesting. Second, it is worthwhile to compare our search proposal against other available timetabling systems on these challenging set of UTPs or other real-life instances of timetabling problems. Lastly, the effects of various heuristic ordering strategies on our proposal or other related heuristic search methods in solving general UTPs should be thoroughly studied.

References

[1] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: an experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research*, 39(3):378 – 406, 1991.

[2] Alvin C.M. Kwan, Ken C.K.Chung, Kammy Yip, Vincent Tam, "An Automated School Timetabling System Using Hybrid Intelligent Techniques", In Proceedings of the 14th Internation Symposium on Methodologies of Intelligent Systems (ISMIS'2003), Japan, October, 2003.

[3] The Meta-Heuristics Network. The International Timetabling Competition 2002 (Oct. 2002 to Mar. 2003) at http://www.idsia.ch/Files/ttcomp2002/.

[4] Steven Minton, Andy Philips, Mark D.Johnston, Philip Laird, "Minimizing Conflicts: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems", Artificial Intelligence, 58, 1992, Pages 161-205.

[5] Peter J. Stuckey, Vincent Tam, "Improving Evolutionary Algorithms for Efficient Constraint Satisfaction", The International Journal on Artificial Intelligence Tools, the World Scientific Publishers, Vol. 8, No. 4, pages 363 - 383, December, 1999.

[6] David Ting, "Intelligent Micro-schedulers for College/University Timetabling", Final-Year Report, Computer Engineering, Dept. of E.E.E., University of Hong Kong, May, 2003.

[7] E. Tsang. *Foundations of Constraint Satisfaction.* Academic Press, 1993.

[8] Masazumi Yoshikawa, Kazuya Kaneko, Yoichiro Nakakuki, "Improving a Heuristic Rapair Method for Large-Scale School Timetabling Problems", CP99, 1999, Pages 275-288.