A Constraint Programming Approach to Solving University Course Timetabling Problem (UCTP)

Article *in* Journal of Computational and Theoretical Nanoscience · November 2017

DOI: 10.1166/asl.2017.10211

3 authors:

Yik Junn Kuan
Universiti of Malaysia Sabah
9 PUBLICATIONS **56** CITATIONS
SEE PROFILE

Joe Henry Obit
Universiti of Malaysia Sabah
48 PUBLICATIONS **765** CITATIONS
SEE PROFILE

Rayner Alfred
Universiti of Malaysia Sabah
227 PUBLICATIONS **2,111** CITATIONS
SEE PROFILE

Advanced Science Letters
Vol. 4, 400–407, 2016

A Constraint Programming Approach to Solve University Course Timetabling Problem (UCTP)

Kuan Yik Junn, Joe Henry Obit, Rayner Alfred
Faculty of Computing and Informatics, Universiti Malaysia Sabah, Sabah, Malaysia

This paper presents the implementation of Constraint Programming in UCTP with special requirement based on the requirement given by Academic Service Divison of UMSLIC, with various forms of constraint which needed to be satisfied in order to obtain a feasible solution for the real-world course timetabling problem obtained from the Universiti Malaysia Sabah Labuan International Campus (UMSLIC), Malaysia. The problem domains in UMSLIC have several constraints that need to be satisfied. Solutions are feasible if all the hard constraints are satisfied. This research does not take into account the soft constraints involved in the domains. The Constraint Programming approach algorithm is tested over three real world datasets: testing dataset; semester 2 session 2014/2015 dataset; semester 1 session 2015/2016 dataset. The result shows that the algorithm studied in this research is able to produce feasible solution within short period of time without violating any hard constraints, which is applicable towards the UCTP in UMSLIC.

Keywords: Constraint Programming, University Course Timetabling.

1. INTRODUCTION

The University Course Timetabling Problem (UCTP) has been studied comprehensively by many researchers using both heuristics and meta-heuristics methods. It requires a heavy task of assigning various events in university such as activities, tutorials, lectures, etc[1]. The benchmark datasets are obtained from the Academic Service Division of UMSLIC in order to model the solutions and determine the hard constraints. The complexity of the characteristics of the problem depends on the policies in the institution itself[3]. The complexity of this problem domain makes it very difficult to find an optimal solution. Hence, a Constraint Programming approach algorithm is studied and proposed in this paper to solve the UCTP.

The research studies the implementation of Constraint Programming in UCTP with special requirement based on the requirement given by Academic Service Divison of UMSLIC, with various constraints needed to be satisfied.

*Email Address: joehenryobit@ums.edu.my

All the constraints obtained are known to be NP-complete2 as most of the smaller problems are linked with additional constraints. This research follows the capacitated category[9] where the room constraint is considered as one of the hard constraints. Therefore, the algorithm performance will be tested in order to cope with the high density conflicts based on the hard constraint. Even though un-capacitated benchmark datasets are commonly used by the researchers, the problem does not represent the real world problem in timetabling domain[1]. In fact, the capacitated problem is more relevant in reflecting towards the real-world problem.

This paper is organized as follows. Section 2 describes the background of the UMSLIC problem domain and the constraints involved which also provides an overview of the relevant timetabling literature. Meanwhile, Section 3 describes the formulation model based on the hard constraints involved and the implementation of the algorithm will be further described in Section 4. The experimental results of Constraint Programming will be discussed in Section 5, whereas Section 6 concludes this

 1936-6612/2011/4/400/008 doi:10.1166/asl.2011.1261

paper by drawing the conclusion and briefly presents the future works.

2. RELATED WORKS

The course enrolment for each student is recorded by the Faculty of Computing and Informatics as well as by the Faculty of International Finance in UMSLIC campus. The record is then forwarded to the Academic Service Division (ASD) for scheduling. However, the scheduling process is done manually by the senior staff members. The process might fail to satisfy all the hard constraints identified, thus it is a challenging task and the task is very time consuming. All the hard and soft constraints thus need to be maximized or minimized while not violating the hard constraints[14]. In this course scheduling problem, the main task is to assign all courses into available timeslots while minimizing the number of unsatisfied hard constraints given. In most situations, the problem becomes more challenging when there are constraints that are conflicted with each other where satisfaction of one constraint might lead to violation of other constraints[13].

However, the constraints specified in the UMSLIC's domain are slightly different from the constraints obtained from the literatures due to different requirements specified in terms of resources. For instance, the 4th hard constraint in UMSLIC which will be discussed in the next section is mostly not mentioned by any literature. In general, UMSLIC consists of 18 available rooms with different capacities and 35 timeslots with each slot occupies 2 hours period. The number of students and courses offered are varied for each semester. Table 1 shows the number of students, courses, timeslots and room for each semester in UMSLIC with the dataset collected from the department of ASD.

Table.1. Summary of datasets

Datasets	No. of Students	No. of Courses	No. of Timeslots	No. of Rooms
Test Set	2104	98	30	12
Sem 2 s2014/15	2248	112	35	18
Sem 1 s2015/2016	2371	125	35	18

That it's very difficult for one institution to have the same constraints with another institution due to different requirements between every institution[12]. Therefore, a course timetabling is built in order to satisfy the requirements stated in this problem domain. Most of the the hard constraints in course timetabling introduced[15] are listed as follow:

a) No student can be assigned into more than one class at the same time
b) The course should be assigned into a room that fulfills all the predefined requirements of the course
c) The number of students attending the course should be less than or equal to the capacity of the room
d) No more than one course is allowed to be assigned into each room given a specific timeslot

3. FORMULATION MODEL

In UMSLIC, the Academic Service Division is responsible for planning and managing the entire academic schedule. Although the schedule produced is feasible, several processes of inspection and checking are performed. The model for instance hardness indicates the possible uses in the hardness of the prediction model in the experiment research[10]. Therefore, the proposed objective in this research is to develop a formal formulation model in order to evaluate the effectiveness of the algorithm. The hard constraints identified in the UMSLIC course timetabling are as follow:

a) No student should attend to more than one class at a time.
b) The total number of students enrolled in each course should be smaller than or equal to the capacity of the room assigned.
c) Not more than one course assigned to a specific room at the same time.
d) Each type of course offered should be assigned to a predefined period of timeslots only.

In this section, the formal model of UMSLIC course timetabling is presented below.

$1, \dots, C \;\; where\; C = total\; number\; of\; courses$
$1, \dots, S \;\; where\; S = total\; number\; of\; students$
$1, \dots, T \; where\; T = total\; number\; of\; timeslots$
$1, \dots, R \;\; where\; R = total\; number\; of\; rooms$

The courses offered in UMSLIC are further divided into four categories. A specific type of course can be only assigned into certain timeslots. The four categories are categorized as follow: Main Courses (MC); Knowledge Enhancement Courses (KEC); Language Courses (LC) and Co-Curricular Courses (CCC). All these courses are accumulated to get the total number of courses, *C*. The model for each category is described as follow:

$1, \dots, c_1 \; (c_1 = total\; number\; of\; MC)$
$1, \dots, c_2 \; (c_2 = total\; number\; of\; KEC)$
$1, \dots, c_3 \; (c_3 = total\; number\; of\; LC)$
$1, \dots, c_4 \; (c_4 = total\; number\; of\; CCC)$
$C = c_1 + c_2 + c_3 + c_4$

The objective is to satisfy all the hard constraints, Hc_1, Hc_2, Hc_3, Hc_4 where the cost function F can be calculated as shown in Equation 1.

$$F = Hc_1 + Hc_2 + Hc_3 + Hc_4 \quad (1)$$

In order to produce a feasible timetable, the total cost function, *F,* must be zero so that none of the hard constraint in Hc_1, Hc_2, Hc_3, Hc_4 are violated. Therefore, it can be summarized that the equation to produce a feasible timetable is as follow:

$$F = Hc_1 + Hc_2 + Hc_3 + Hc_4 = 0 \quad (2)$$

For hard constraints, the given penalty cost (represented as $\lambda_1 \, through \, \lambda_4$) are usually set up to a very high value, with the value of ∞. In this case, the penalty cost for each hard constraint is initialized to a value of 100,000 so that the cost function *F* can be easily used to identify if any of the hard constraint is not satisfied. The penalty for every weight of hard constraint is summarized in the Table 2.

Table.2. Penalty assignment for constraint violations

Weight	Penalty	Description
λ_1	100,000	Hard constraint 1, Hc_1
λ_2	100,000	Hard constraint 2, Hc_2
λ_3	100,000	Hard constraint 3, Hc_3
λ_4	100,000	Hard constraint 4, Hc_4

In the case of scheduling, the cost calculation both shows the factor of influence towards one another. The penalty costs for soft constraints are presented as follow:

For the constraint 1 shown in equation (3), student attends more than one lecture at the same time is given by

$$H_{C_1} = \lambda_1 \sum_{i=1}^{C-1} \sum_{j=i+1}^{C} CC_{ij} \quad (3)$$

where *C* is the total number of courses, λ_1 is the weight, CC_{ij} is the number of students who have at least two classes *i* and *j* at the same time.

For the hard constraint 2 shown in equation (4), the size of the course enrolment should be smaller than or equal to the capacity of the assigned room is given by

$$H_{C_2} = \lambda_2 \sum_{i=1}^{C} \sum_{j=1}^{R} CR_{ij} \quad (4)$$

where *C* is the number of courses, *R* is the number of rooms, λ_2 is the weight, CR_{ij} is the number of rooms which the course *i* larger than the capacity of that room *j*.

For the hard constraint 3 shown in equation (5), the condition of not more than one course is assigned to a particular timeslot and room at the same time is given by

$$H_{C_3} = \lambda_3 \sum_{i=1}^{R} \sum_{j=1}^{T} RT_{ij} \quad (5)$$

where *R* is the number of rooms, *T* is the number of timeslots λ_3 is the weight, RT_{ij} is the number of courses which have different lectures, room *i* and timeslot *j* at the same time and classroom.

For the hard constraint 4 shown in equation (6), the type of course should be assigned into a specific timeslot is given by

$$H_{C_4} = \lambda_4 \left(\sum_{i=1}^{c_1} \sum_{t=1}^{T} M_{it} + \sum_{j=1}^{c_2} \sum_{t=1}^{T} P_{jt} + \sum_{k=1}^{c_3} \sum_{t=1}^{T} L_{kt} + \sum_{l=1}^{c_4} \sum_{t=1}^{T} C_{lt} \right) \quad (6)$$

where c_1, c_2, c_3, c_4 are the total number of main courses, knowledge enhancement courses, language courses, and co-curricular courses respectively, λ_4 is the weight, *T* is the number of timeslot and M_{it}, P_{jt}, L_{kt} and C_{lt} are violation of the particular course *i, j, k, l* to be assigned in timeslot *t*.

Equation (3) is applied to solve the hard constraint of conflict schedule, which imposes the degree between course conflicts among one another. Meanwhile, equation (4) is used to identify the constraint in terms of course size to fit into the size of the room. Equation (5) identifies the constraint of courses held together in the same room while equation (6) selects the specific type of courses which can be only assigned into specific timeslot. The total cost function *F*, from equation (1) determines the penalty cost based on the university timetabling. In order to produce a feasible solution, the cost function *F* must be zero as shown in equation (2) or else the timetable is considered as a non-feasible solution. The algorithms implemented will be further described in details in the following section.

4. CONSTRAINT PROGRAMMING

Constraint Programming was previously experimented by Jaffar and Lassez in their research to solve the hard constraints in the domain[8]. The experimental design was setup to perform the searching for feasible solution by using a Constraint Programming. The Constraint Programming identifies the hard constraint through binary matrices or diagonal matrices. As there are four hard constraints identified in UMSLIC problem, each of the constraint is constructed in binary matrices which represented below:

Equation 7 shows the course conflict matrix, CC_{ij} which identifies the conflict between courses or otherwise

$$CC_{ij} \begin{cases} 1 & if \; course \; i \; clashes \; with \; course \; j \\ 0 & otherwise \end{cases} \quad (7)$$

Equation 8 shows the course room matrix, CR_{ir} which identifies a particular course larger than a particular room or otherwise

$$CR_{ir} \begin{cases} 1 & if \; course \; i \; larger \; than \; room \; r \\ 0 & otherwise \end{cases} \quad (8)$$

Equation 9 shows the room timeslot matrix, RT_{ij} which identifies a specific room and timeslot are assigned with a course or otherwise

$$RT_{rt} \begin{cases} c & if \; course \; c \; is \; assigned \; in \; timeslot \; t \; and \; room \; r \\ -1 & otherwise \end{cases} \quad (9)$$

Equation 10 shows the course timeslot matrix, CT_{ij} which identifies whether a specific type of course is restricted to a certain timeslot or otherwise

1936-6612/2011/4/400/008 doi:10.1166/asl.2011.1261

$$CT_{it} \begin{cases} 1 & if\ course\ i\ is\ assigned\ into\ timeslot\ t \\ 0 & otherwise \end{cases} \quad (10)$$

By using the matrices explained above, the constraint programming search for a feasible solution within a limited search space and begin assigning the course if the solution space is feasible.

Technically, the algorithm involves two phases of assigning all the courses into the available and feasible resources. Three pools of solutions are used as allocation of resources which divided into *poolOfUnscheduledCourses* (PUC) that refers to the courses which have not been scheduled, *poolOfScheduledCourses* (PSC) that refers to the courses which have successfully scheduled and *poolOfFailedScheduledCourses* (PFSC) that refers to the courses which could not find a feasible solution in the available resources or failed to be scheduled. The algorithm skips the second phase if the first phase has successfully assigned all the courses into the resources or into the PSC which contains all the courses in timetable.

In the first phase, the algorithm selects a course, timeslot and a room at random from the PUC. If all the conditions are met, the selected course will be assigned into the particular timeslot and room or else the selected timeslot and room will be randomized again in order to determine the next feasible solution which can be assigned into the new timeslot and room. The course is then moved from PUC to the PSC, if it is successfully assigned while unsuccessful assignment will result in moving it into the PFSC.

The experiment is then continued with the next iterative procedure if phase 1 failed to schedule all the courses. This process involved the disturbing of solution in PSC. Courses are selected at random from the scheduled courses pool and the assignment of timeslot and room will be adjusted. Then, the course obtained from PFSC will then be assigned to the timeslot and room freed from the courses that are successfully assigned. The course is then removed from the PFSC after it is successfully assigned with a particular timeslot and room, and also added to the PSC.

5. EXPERIMENTAL SETUP AND RESULTS

The experiment is conducted using three datasets of real world instance which are obtained from Academic Service Division in UMSLIC and with 50 iterations for each datasets in order to evaluate the proposed constraints programming approach in solving the University Course Timetabling Problem (UCTP). The set of experiment procedure have been done by many researchers[5] in order to investigate the algorithm using constraint based reasoning methods which have proven successful.

The results shown in Table 3 indicate that the Constraint Programming algorithm is able to produce feasible solutions in all the 50 iterations. The average cost function for all three datasets are zero which satisfies equation (2) as stated in the previous section as a requirement to produce a feasible timetable. Nonetheless, the quality of the solutions are varies as the algorithm does not perform the movement to improve the soft constraints. Therefore, the cost function can be ignored as long as the hard constraint is always zero.

Table.3. Experimental Results for Constraint Programming

Experiments	Testing Dataset	Sem 2 s2014/2015	Sem 1 s2015/2016
Average cost function, *F*	0	0	0
Average time, s	5.58	1.53	1.10
Max time, s	11.95	5.32	2.30
Min time, s	0.70	0.59	0.61
Standard deviation	4.258	0.980	0.302

In this experiment, the only factor that should be measured between the datasets is the time taken to produce a feasible solution. Testing dataset takes much longer time in average to produce a feasible solution as shown in Table 3. This may be due to the set of constraints in testing dataset which slightly increases the duration of the time in order to find a feasible solution. Meanwhile, semester 2 session 2014/2015 dataset and semester 1 session 2015/2016 dataset has almost the same time to produce a feasible solution in average with 1.53 seconds and 1.1 seconds respectively.

The standard deviation in Table 3 measures the dispersion or consistency in time taken by CP algorithm to produce feasible solution in all 3 datasets. The lower the standard deviation, the better the consistency of the time taken by CP algorithm to solves the timetabling problem. Testing instance produce the lowest consistency (or highest standard deviation) compare with other two datasets. It can be assumed that the constraint in testing dataset is more random due to constraint limitation. Nonetheless, the CP algorithm still manages to produce feasible solutions to all three datasets within short period of time.

6. CONCLUSIONS

Many solution approaches have been proposed in the research field in order to solve the University Course Timetabling Problem (UCTP). The proposed solution in this research which use Constraint Programming algorithm manage to solve the timetabling problem in UMSLIC, which produce a feasible timetable in less than a minute. The measure of effectiveness in CP algorithm is evaluated by the time taken to produce a feasible solution. It can be summarized that all three datasets are able to be scheduled by CP within short period of time.

This research is referred to the heuristics approach[11] in timetabling problem, where generating initial timetables should be implemented in order to solve the hard constrains. The research can be proved satisfying even with Swarm Particle Optimization to solve the hard constraint in short period of time[6,7].

Even hybrid methods[4] can be experimented in UMSLIC domain as future research and the result can be compared with existing experimented algorithm in

UMSLIC in order to determine the best approach to be adapted into this domain.

REFERENCES

[1] Abdullah, S., Shaker, K., McCollum, B., & McMullan, P. (2009, July). Construction of course timetables based on great deluge and tabu search. In *Proceedings of MIC 2009: VIII Metaheuristic International Conference* (pp. 13-16).

[2] Aycan, E., & Ayav, T. (2009, March). Solving the course scheduling problem using simulated annealing. In *Advance Computing Conference, 2009. IACC 2009. IEEE International* (pp. 462-466). IEEE.

[3] Cerdeira-Pena, A., Carpente, L., Farina, A., & Seco, D. (2008, October). New approaches for the school timetabling problem. In *Artificial Intelligence, 2008. MICAI'08. Seventh Mexican International Conference on* (pp. 261-267). IEEE.

[4] Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, *9*(5), 403-432.

[5] Fen, H. S., Deris, I. S., & Hashim, S. Z. (2009). Investigating constraint based reasoning for university timetabling problems. In *International MultiConference of Engineers and Computer Scientists* (Vol. 1).

[6] Ho, S. F. (2009). Incorporating Of Constraint-Based Reasoning Into Particle Swarm Optimization For University Timetabling Problem. *Computer Science Letters*, *1*(1).

[7] Irene, H. S. F., Deris, S., & Hashim, S. Z. M. (2009, June). University course timetable planning using hybrid particle swarm optimization. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation* (pp. 239-246). ACM.

[8] Jaffar, J., & Lassez, J. L. (1987, October). Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages* (pp. 111-119). ACM.

[9] Kahar, M. N. M., & Kendall, G. (2010). The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution. *European Journal of Operational Research*, *207*(2), 557-565.

[10] Kostuch, P., & Socha, K. (2004). Hardness prediction for the university course timetabling problem. In *Evolutionary Computation in Combinatorial Optimization* (pp. 135-144). Springer Berlin Heidelberg.

[11] Landa-Silva, D., & Obit, J. H. (2011). Comparing Hybrid Constructive Heuristics for University Course Timetabling. *Strathprints Institutional Repository*, 224.

[12] Murray, K., Müller, T., & Rudová, H. (2006). Modeling and solution of a complex university course timetabling problem. In *Practice and theory of automated timetabling VI* (pp. 189-209). Springer Berlin Heidelberg.

[13] Qu, R., Burke, E. K., McCollum, B., Merlot, L. T., & Lee, S. Y. (2009). A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling*, *12*(1), 55-89.

[14] Schaerf, A. (1999). A survey of automated timetabling. *Artificial intelligence review*, *13*(2), 87-127.

[15] Socha, K., Knowles, J., & Sampels, M. (2002). A max-min ant system for the university course timetabling problem. In *Ant algorithms* (pp. 1-13). Springer Berlin Heidelberg.Schaerf, A. (1999). A survey of automated timetabling. *Artificial intelligence review*, *13*(2), 87-127.

 1936-6612/2011/4/400/008 doi:10.1166/asl.2011.1261