International Journal of Research
Available at https://journals.pen2print.org/index.php/ijr/
e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 06 Issue 09
August 2019

Constructing University Timetable using CSP Model

Nwe Nwe Hlaing, Mya Thet Nyo
nwe2hlaing@gmail.com, myatthetnyo05@gmail.com

Abstract:

Timetable problem is a well-known multidimensional, constraint assignment problem that focuses in the assignment of courses to faculty members in classrooms within limited time slots. Hence, it is a challenging time-consuming problem facing universities and it belongs to the NP-hard class of problems. In particular, universities regularly need an optimal solution for the course timetable problem. However, a manual solution to this problem by considering all constraints usually requires a long time and hard work to offer proper, optimized solution. Specifically, timetabling problem scan be modeled as Constraint satisfaction problems (CSPs), which are combinatorial in nature. This paper introduces a model that applies the timetabling as a CSP model. The system is able to produce a time table for UCS(Mandalay). The proposed model is tested against real data obtained from UCS (Mandalay) that has a so sophisticated timetable with much interlaced, limited resources and limited time slots.

Keywords

Constraint Satisfaction Problem (CSP) , university timetabling (UTT) ,course timetabling (CTT),

Introduction

The timetabling problem consists of a set of subjects to be scheduled in different timeslots, a set of rooms in which the subjects can take place, a set of students who attend the subjects, and a set of subjects satisfied by rooms and required by timeslots. The heart of the problem is the constraints that exist as regulations within each resource and between resources. There are various solution approaches to solve the timetabling problem. This paper focuses on developing a constraint satisfaction problem model for a university timetabling problem. A solution of a constraint satisfaction problem is a consistent assignment of all variables to values in such a way that all constraints are satisfied.

Indeed, timetabling is a multidimensional assignment problem, which needs to be solved regularly at educational institutions. It is the assignment of courses to faculty members and the assignment of these courses to classroom and time slots [1] in a way that makes optimal use of the available resources [2]. Such a timetable must satisfy certain constraints such as no single teacher teaches more than one class at the same time, no single room is allocated for more than one class at the same time, and so on. Further, it may try to achieve certain objectives such as maximum utilization of classrooms, assigning teachers to his or her preferred courses, etc. Practically, a typical university timetabling problem may comprise thousands of courses, thousands of students, hundreds of instructors, hundreds of classrooms and other resources.

Moreover, timetabling problem has been classified as NP-hard optimization problem (i.e., no polynomial time algorithm is known to solve the problem) [3], meaning that if all combinations were to be examined, the time to solution for reasonable problems would rise dramatically. Therefore, in order to find optimal solutions to such problem, it is necessary to consider all possible solutions to choose the best one that satisfy a wide range of constraints, preferences, and participants and it must be solved in reasonable time.

Although, the university timetabling (UTT) is a major, regular and complex administrative activity in most academic institutions [3], only a few organizations possess reliable automated timetable solvers, and fewer still possess solvers that require no manual intervention. However, most institutions employ the knowledge and experience of expert personnel with regard to the production of good timetable that satisfy all given requirements. Categorically, UTT problems can be classified into two main categories: course timetabling (CTT) problems and examination timetabling (ETT) problems, each with its own sets of constraints and requirements. The focus of this paper is on the CTT. Therefore, lots of research has been invested in order to provide automated support for solving a real-world timetabling problem. Contributions come from the fields of operations research (e.g., graph coloring, network flow techniques) and artificial intelligence (e.g., simulated annealing, tabu search, genetic algorithms and constraint satisfaction [4]). Barták et al. [5] surveyed the main definitions and techniques of constraint satisfaction, planning and scheduling from the Artificial Intelligence point of view.

Related Work

Regarding timetabling problems, many works have been done to accomplish a good solution to their using different operation research (OR) and artificial intelligence (AI) approaches since fiftieth. A large number of problems in AI and other areas of

computer science can be viewed as special cases of the CSP [10]. In addition, many problems in OR fall within the general framework of CSP [8]. The development of effective solution techniques for CSPs is an important research problem. Timetabling problems are considered as combinatorial problems in OR, which can be modeled as CSPs. Researchers in AI usually assume constraint satisfaction approaches as their preferred methods when undertaking such problems [8].A variety of approaches can be used to process the CSPs. For instance, LP techniques can be applied to find an exact solution. On the other hand, there are various approaches provide an approximate solution, including local search methods and neural networks as a special purpose technique used for solving CSPs.

There are many research works provide the usage of constraint-based approaches. In these methods, the events of a timetabling problem are modeled by a set of variables to which values have to be assigned subject to a number of constraints[11]. When the propagation of the assigned values leads to an infeasible solution, abacktracking process enables the reassignment of value(s) until a solution is found that satisfies all of the constraints. To begin with, Abdennadher and Marte [12]showed how to model the timetabling problem as a partial constraint satisfaction problem and gave a concise finite domain solver implemented with Constraint Handling Rules that allows for making soft constraints an active part of the problem solving process. They improved efficiency by reusing parts of the previous year's timetable. Zhang and Lau [13] developed a CSP model for a university timetabling problem. They investigated a sample case study problem and implemented an approach for constraint satisfaction programming using ILOG Scheduler and ILOG Solver. They used various goals in ILOG to investigate the performance of the CSP approach. Further, Ojugo et al. [14]build constraint satisfaction models to search the space for the timetable scheduling state and satisfies all constraints and criteria that guarantee the reasoning process through an explicit structure that conveys data about the problem. They aimed to find a complete assignment that satisfies certain constraints to yield a valid schedule. In addition, they provided a study that surveyed NP-complete task of academic timetabling at the University of Benin, Nigeria and they adopted a rule-based expert system to yield an initial solution for the models. They showed that their models yielded a valid schedule for the University of Benin in Nigeria considering student preference, medium constraints of high priority.

At the same time, there are various approaches applied to the CTT problem including Linear Programming (LP). LP is widely used for the solution of timetabling problems, like Ribi and Konjicija[15], Wattanamano, Thongsanit and Hongsuwan[1] and Czibula et al. [16]. Other approaches have been recently applied to solve the CTT problem including heuristics and meta-heuristic, graph coloring, network flows, genetic algorithms and other OR and/or AI methods [17].

The approaches to solve timetabling problem

The timetabling problem as a special case of scheduling: "Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives." Timetabling problem is generally considered as a resource allocation problem in Operations Research, where resources of lecturers, students, classrooms and subjects are to be allocated into timeslots of a weekly timetable to achieve an objective function subject to constraints among resources [18].

Timetabling problems is a type of assignment problems with large amount of complex constraints, thus usually can be easily modeled as constraint satisfaction problems (CSP) [17]. The application for solving the timetabling problem using constraint satisfaction programming approach allows the formulation of all the constraints of the problem in a more declarative way than other approaches [13, 14].

Thus the CSP is particularly well suited for timetabling problems, since it allows the formulation of all constraints of the problem in a more declarative way than other approaches. Constraint satisfaction problem (CSP) deals with assignment of values from its domains to each variable such that no constraint is violated [18, 19]. CSP has three components: variables, values and constraints. In general, CSP consists of: a finite set of variable X = {x1,…,xn} with respective domains D = {D1,…, n} which list the possible values for each variable Di = {vi,…,vk} and a set of constraints C = {C1, …, Ct} [19, p.31]. The constraints limit the possible values that a variable can have. A solution of a CSP is a consistent assignment of all variables to values in such a way that all the constraints are satisfied.

There are two approaches to solving CSP. One is using the search algorithms and the other is using the consistency technique. Consistency techniques have been widely studied to simplify constraint network before or during the search of solutions. Dechter [19] defines arc-consistency as a process that ensures any valid value in the domain of a single variable has a valid match in the domain of any other variables in the problem. Arc (Vi,Vj)is arc consistent if for every value x in the current domain of Vi there is some value y in the domain of Vj such that Vi=x and Vj=y is permitted by the binary constraint between Vi and

Vj. The concept of arc-consistency is directional. If the process involves three variables then it is known as path consistency. In general a graph is k-consistent if there exists (k-1) variables that satisfy all the constraints among these variables and there also exists a value for this kth variable that satisfies all the constraints among these k variables [19].

Most algorithms for solving the CSP search systematically through the possible assignments of values to variables. Such algorithms are guaranteed to find a solution if one exists or to prove that the problem has no solution, but this process may take a very long time. Backtracking is the most common method for performing systematic search. In the backtracking algorithm, the current variable is assigned a value from its domain. This assignment is then checked against the current partial solution. If any of the constraints between this variable and the last variables is violated, the assignment is abandoned and another value for the current variable is selected [19].

There are three disadvantages of backtracking approach: thrashing, redundant work and late detection of conflict [10]. Thus look-ahead scheme is proposed to overcome some or all of these problems. The look ahead scheme is invoked whenever the algorithm is preparing to assign a value to the next variable [19].

There are two approaches in the look ahead scheme. The first approach is called forward checking. This approach checks only the constraints between the current variable and the future variables. When a value is assigned to the current variable, any value in the domain of a future variable, which results in conflicts with this assignment, is removed from the domain. This means if the domain of the future variable is empty, it infers that the current partial solution is inconsistent and another value should be tried or it should backtrack to the previous variable [20]. The second approach is called (full) look ahead or maintaining arc-consistency. This is an approach that uses full arc-consistency during the look ahead scheme. It allows branches of the search tree that will lead to failure to be pruned earlier [20].

Look back schemes are invoked when the algorithm encounters a dead-end and prepares for the backtracking step [19]. All look back schemas share the disadvantage of late detection of the conflict. It solves the inconsistency when it occurs but does not prevent the inconsistency from occurring. There are two approaches to look back scheme: backjumping and backmarking. Backjumping works the same way as backtracking. The difference is during the backtracking step. In backjumping, it analyses the situation in order to identify the source of inconsistency. Backjumping backtracks to the most recent conflicting variable, whereas backtracking backtracks to the immediate past variable [20]. In backmarking, it avoids redundant constraint checking by recording the highest level that is last backtracked to. This helps to reduce repetitive consistency checking by remembering the success and failure of compatibility checks, which have already been performed [19].

CSP model for timetabling problem

Concerning CSP, it is defined (in [10, 8,]) by a finite set of variables, each of which has a finite domain of values, and a set of constraints. Each constraint is defined over some subset of the original set of variables and restricts the values these variables can simultaneously take. We develop our model with the most common definition and syntax of CSP. A CSP is defined as three sets: X, D, and C, where:

- X = {x1, … ,xn} is the set of variables called domain variables;
- D = {D1, … ,Dn} is the set of domains(the possible values for the corresponding variable);
- C = {C1, … ,Cc} is the set of constraints (relations defined on a subset of all variables).

A simple backtracking algorithm for constraint satisfaction problems is shown in figure 1. The term backtracking search BACKTRACKING is used for a depth-first search that chooses values for SEARCH. one variable at a time and backtracks when a variable has no legal values left to assign. The algorithm is modeled on the recursive depth-first search. By varying the functions SELECT-UNASSIGNED-VARIABLE and ORDER-DOMAIN-VALUES, we can implement the general-purpose heuristics. The function INFERENCE can optionally be used to impose arc-, path-, or *k*-consistency, as desired. If a value choice leads to failure (noticed either by INFERENCE or by BACKTRACK), then value assignments (including those made by INFERENCE) are removed from the current assignment and a new value is tried.

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
return BACKTRACK({ }, csp)
function BACKTRACK(assignment, csp) returns a solution, or failure
if assignment is complete then return assignment
var←SELECT-UNASSIGNED-VARIABLE(csp)
for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
if value is consistent with assignment then
add {var = value} to assignment
inferences ←INFERENCE(csp, var , value)
if inferences = failure then
add inferences to assignment
result ←BACKTRACK(assignment, csp)
if result = failure then return result
```

remove {var = value} and inferences from assignment
return failure

Figure 1. Backtracking Algorithm

Proposed Model of CTT

The problem consists of scheduling a set of classes in different timeslots subject to satisfying the following constraint: no student attends more than one class at the same time, the room must be big enough for all the attending students, and only one class is scheduled in one room at any one timeslot.

Let S = {s1, s2,…sn} be the set of sections(1cst-A,1cst-B,2cs,3cs,…etc);

L= {l1, l2,…lo} be the set of subjects taught ;

Ts= {ts1, ts2,…, tsm } be the available teaching periods(time slot) ;

T={t1,t2,…..,tp} be the set of teacher;

R = {r1, r2, …, rq} be the set of rooms available.

A feasible timetable is one in which all events have been assigned a timeslot and a room so that the following constraints are satisfied:

C1: A teacher cannot be taught in two classes at the same time, so avoid clashing the course of teacher.

C2: One room should not be assigned to more than one subject for the same timeslot.

C3: Certain timeslots are reserved for lab hours and activities.

C4: Subject cannot be assigned more than 4 timeslots (including lab time).

The model we propose for a timetabling problem as a CSP is as follows: a timetable is a constrained variable the value of which is a function associating a value to each slot in time t. The timetable item is given by the set of subjects. Note that the subject can be offered as a lecture or a tutorial, which is considered as a timetable item. Basically our task consists in instantiation of the set of four tuples CSP (section, lecture, timeslot, teacher), i.e., each lecture or subject taught by teacher has assigned its set of section and time.

Implementation

In solving UTT problem, we define each section timetable variables and values. Time slots as variables and assigning subjects as values. The backtracking algorithm is applied for scheduling timetable. Whenever variables(time slot) is assigned to order domain value subject heuristic function is applied for checking constraint violations. If the failure is detected, the assigning process backtracks again.

The solution for a feasible timetable for the proposed model is resulted in the two-dimensional array that is one dimension (row) for days and another dimension (column) for time slots. The result time table would be saved as Excel file shown in figure 2.

Second Year(2CST-A)

Days	TS1	TS2	TS3	TS4	TS5	TS6	TS7
Mon	Tutorial	201	204	203	205	202	Eng
Tue	Tutorial	205	204	205	Eng	206	202
Wed	203	202	205	Lab(201)	Lab(201)	206	Library
Thurs	203	204	205	201	206	205	Eng
Fri	Activity	Activity	202	206	Lab(203)	204	Eng

Figure 2. Exported excel file for Second Year Time table

Conclusion

The proposed model for the university timetable is based on finite domain technique for CSP. Our system solves the problem efficiently. The development time for the program is much shorter than time spending on the manual approach. Our experience shows that CSP model is a practically viable means for timetable scheduling in a university. The results from running our model of the CTT problem conclude and prove that all the required constraints are successfully verified. Moreover, an important issue is appeared in the result timetable, our model balance the teaching load for teachers that the college faculties For further work, we are able to enrich this model at the UCS(Mandalay) to build the university CTT.

Acknowledgements

I would like to greatly thank Daw Aye Aye Khaing, Associate Professor and Head of English Department and Daw Yu Yu Hlaing, Associate Professor of English Department for checking language. Contribution of others who might have given suggestions or review comments.

References

[1] R. Wattanamano, K. Thongsanit and P. Hongsuwan, "The Development of Mathematical Model for a University Course Timetabling Problem," Silpakorn University Science and Technology Journal, vol. 5, no. 2, pp. 46-52, 2011.

[2] K. Kumar, Sikander, R. Sharma and Kaushal, "Genetic Algorithm Approach to Automate University Timetable," International Journal of Technical Research(IJTR), vol. 1, no. 1, 2012.

[3] A. O. Adewumi, B. A. Sawyerr and M. M. Ali, "A heuristic solution to the university timetabling problem," Engineering Computations: International Journal for Computer-Aided Engineering and Software, vol. 26, no. 8, pp. 972-984, 2009.

[4] S. Abdennadher and M. Marte, "University Course Timetabling using Constraint Handling Rules," Journal of Applied Artificial Intelligence, vol. 14, no. 4, pp. 311-325, 2000.
[5] R. Barták, M. A. Salido and F. Rossi, "Constraint satisfaction techniques in planning and scheduling," Intelligent Manufacturing, vol. 21, no. 1, pp. 5-15, 2010.
[6] F. Rossi, P. van Beek and T. Walsh, Handbook of Constraint Programming, 1 ed., Elsevier Science, 2006, p. 978.
[7] A. Schaerf, "A Survey of Automated Timetabling," Artificial Intelligence Review, vol. 13, no. 2, pp. 87-127, 1999.
[8] S. C. Brailsford, C. N. Potts and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," European Journal of Operational Research, vol. 119, pp. 557-581, 1999.
[9] S. Abdennadher and M. Marte, "University Course Timetabling using Constraint Handling Rules," Journal of Applied Artificial Intelligence, vol. 14, no. 4, pp. 311-325, 2000.
[10] V. Kumar, "Algorithms for Constraint-Satisfaction Problems: A Survey," AI Magazine, vol. 13, no. 1, pp. 32-44, 1992.
[11] S. Petrovic and E. Burke, "University timetabling," in Handbook of scheduling: algorithms, models, and performance analysis, Chapman and Hall/CRC, 2004, pp. 1-23.
[12] S. Abdennadher and M. Marte, "University timetabling using constraint handling rules," JFPLC, pp. 39-50, 1998.
[13] L. Zhang and S. K. Lau, "onstructing university timetable using constraint satisfaction programming approach," Computational Intelligence for Modelling, Control and Automation, vol. 2, pp. 55-60, 2005.
[14] A. Ojugo, I. Iyawa., F. Aghware., M. Yerokun and E. Ugboh, "Comparative Study of the Timetable Constraint Satisfaction Problem," in 5th International Conference on Cir-cuits, Systems, Control, Signals (CSCS '14), Salerno, Italy, 2014.
[15] S. Ribi and S. Konjicija, "A Two Phase Integer Linear Programming Approach to Solving the School Timetable Problem," in 32nd International Conference on Information Technology Interfaces, Cavtat, Croatia, June 21-24, 2010.
[16] O. Czibula, H. Gu, A. Russell and Y. Zinder, "A Multi-Stage IP-Based Heuristic for Class Timetabling and Trainer Rostering," in 10th International Conference of the Practice and Theory of Automated Timetabling, York, United Kingdom, 2014.
[17] K. Zervoudakis and P. Stamatopoulos, "A Generic Object-Oriented Constraint-Based Model for University Course Timetabling," in Practice and Theory of Automated Timetabling III, vol. 2079, Springer Berlin Heidelberg, 2001, pp. 28-47.
[18] Nadel, B. A., 1989, Constraint Satisfaction Algorithms, Computational Intelligence, vol.5, pp.188-224.
[19] Dechter, R., 2003, Constraint Processing, Morgan Kaufmann.
[20] Barták, R., 1998, On-line Guide to Constraint Programming, http://kti.ms.mff.cuni.cz/~bartak/constraints/ (Access: 25 March, 2004).