Journal of Harbin Engineering University
ISSN: 1006-7043
Vol 45 No. 9
September 2024

A Distributed Approach for Scheduling Classes at Educational Institute

[1]Ranjan Kumar Thakur, [2]Nawin Kumar Agrawal, [3]Pankaj Kumar
[1]Department of Mathematics, Maharani Kalyani College, Darbhanga, Bihar, India - 846003
[2]Department of Mathematics, Lalit Narayan Mithila University, Darbhanga, Bihar, India - 846004
[3]Department of Mathematics and Scientific Computing, NIT Hamirpur, Himachal Pradesh, India - 177005

Abstract: Academic experiments are being conducted at a rapid rate, making time table scheduling automation in educational institutions imperative. These frequently updated, strategically planned teaching and learning frameworks must be applied at the institutional level, which requires institutions to alter their outdated scheduling procedures. While manual scheduling is feasible, it requires hours of continuous concentration to verify a multitude of limitations, which is inhumane. It is far more laborious to modify the scheduled classes than it is to do rescheduling. The Constraint Satisfaction Problem (CSP) technique is a logical choice for scheduling automation. It could be scheduled more quickly using a Distributed Constraint Satisfaction Problem (DCSP) method. Given that the current scheduling work has multiple objectives as well as distributed in nature, the DCSP technique is the preferable option. In order to map the problem as a DCSP, we first address it as a distributed problem in this study. An asynchronous backtracking technique is then used to reach to a solution.
Keywords: Distributed constraint scheduling, Timetable scheduling, DCSP, scheduling automation.

1. Introduction

All over the world, the academicians try to introduce new way to deal with the recent changes by designing new policies in which new courses are incorporated or the whole course structures is redesigned. For example, the Indian government is currently working on implementing National Education Policy (NEP) 2020. What we want to convey is that the curriculum of different universities and colleges in India as well as all over the world changes on regular basis. The bulk of college timetable schedulers that are accessible are based on the curricula of universities and other institutions located outside of India. Many modifications must be made to those schedulers in order for them to be used by Indian colleges. This research develops a method that takes into consideration the requirements and limitations that were often present for colleges of Indian universities. University/Colleges offer different courses. The curriculum of these courses is divided into semesters or years. In a particular semester/year, the curriculum consists of a number of papers. These papers are distributed among different departments of the college/university. By scheduling classes, we mean to assign specific class room and specific faculty members to these papers in a specific time on a specific day of a week (timeslot).

Scheduling classes requires checking a variety of constraints. It is crucial to make sure that separate papers are not given the same classroom time while arranging the lectures. Additionally, it is necessary to confirm that a faculty member's two lectures are not scheduled for the same classroom time. With the introduction of NEP 2020, some papers are designed to be opted by more than one stream of students. There are some mathematics papers that can be opted by economics or commerce or language students. This will add on the complexity of scheduling as we have to take care of the fact that when that mathematics paper is being taught at the same time the students who opted the paper must not be assigned to any class. We term such papers as Generic Papers (GP). Increase in the number of GP will complicate the scheduling process.

The number of lectures that can be assigned to a specific paper in a week and the number of lectures that can be assigned to a faculty member based on their designation are both limited in the designed curriculum by the rules and regulations set up by the competent authority. Suppose a paper requires running 5 lectures per week in a semester/year then the workload of that paper is 5. Similarly, if the competent authority sets a limit that a specific designated faculty should take at least 16 lectures

per week then 16 is the workload of that designation. Based on the number of papers and the lecture requirement of each paper, one department calculates its workload. Faculty requirement for colleges/universities can be met with ease but classrooms are scarce resource and required to be used optimally.

Problem Statement:

Given the details of Papers to be taught in an academic session, the available faculties and classrooms, the problem is to design an automation framework to be implemented by object-oriented programming language that can schedule classes and faculties to these papers satisfying the following constraints:

1. Two faculties should not be assigned a same timeslot, other than practical papers.
2. A faculty should not be assigned to more than one papers at the same time.
3. When a Generic Papers (GP) runs, no other papers (for the students who chooses that GP) should run.
4. More than one lecture of a paper if assigned in a day then those lectures should be continuous lectures.
5. Two papers should not be assigned the same timeslot

2. Related Works

The above poised problem is not new. Scholars seeking to discover practical answers are typically drawn to the complicated, multiobjective, and extremely constrained nature of educational scheduling. In 1997, Carter and Laporte [1] published research titled" Recent Development in Practical Course Timetabling.", where various approaches from 1986 to 1996 were recognized. A constraint programming technique is used by Valouxis and Housos (2002) to identify a solution in their study [2], and local search is used to further improve their findings. A two-phase hybrid technique was presented by Jat and Yang (2011) to address the challenge of university course scheduling [3]. The Guided Search Genetic Algorithm was utilized in the first phase, while the Tabu Search heuristic was employed in the second. In 2015, Hakan Andersson [4] introduced an automation solution for the school scheduling issue. Two meta heuristic methods, Tabu Search and Simulated Annealing, with previously successful outcomes, are applied in his thesis. Practices in Timetabling in Higher Education Institutions, a 2017 study [5] by Oude Vrielink, Jansen, Hans, and van Hillegersberg offers a survey of the various timetabling literatures by highlighting the variations and commonalities in theory and practice. For their respective timetabling problems, a number of other approaches are also used, including fuzzy multiple heuristic approach [6], hyperheuristics approaches [7], fuzzy multiple heuristic approach [8] etc. A DCSP approach to allocate nurses to different departments at different timings in a large hospital was investigated by Gadi Solotorevsky and Ehud Gudes [9].

3. Methodology

Constraint Satisfaction Problem (CSP):

A constraint satisfaction problem (CSP) is a triplet (X, D, C) where $X = \{x_1, x_2, \dots, x_n\}$ is a non-empty finite set of variables, $D = \{D_1, D_2, \dots, D_n\}$ is the non-empty set of non-empty domains for the variables present in X and C is the set of constraints restricting the values that the variables can simultaneously take in a problem. A solution of CSP is an assignment to the variables presents in X from their respective domains satisfying all the constraints present in C.

Distributed Constraint Satisfaction Problem (DCSP):

In a Distributed Constraint satisfaction Problem (DCSP), variables and constraints are distributed among autonomous agents. To solve the DCSP various techniques have been devised. We use asynchronous backtracking to solve DCSP.

Asynchronous backtracking to solve DCSP:

In a DCSP, variables and constraints are distributed among autonomous agents. Here, it is to be noted that variables of more than one agent may be present in a constraint. As constraints are distributed among agents, an agent may or may not have the knowledge of constraints which involves variables present in it. To evaluate a constraint by an agent it must have the knowledge of the assignment of each of the variable involved in the constraint. Since agent can only assign values to the variables within itself, to evaluate a constraint, it must have the knowledge of values of other variables involved in it. To have the knowledge of these values, a mode of communication is needed

among agents where agents could share information among themselves. Thus, there are two ways an agent can be viewed one where it evaluates constraints involved in it (viewed as constraint evaluating agent) and another where it sends values of variables present in it to other agent (viewed as value sending agent). An agent can be both a value sending agent to other constraint evaluating agent as well as constraint evaluating agent that receives values from other agents.

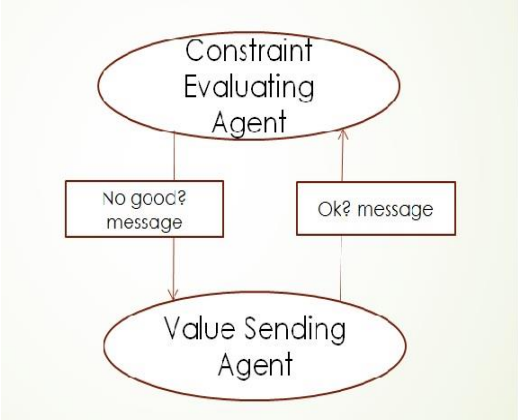

Figure 1: Two-way interaction between agents

Since, there is a need to establish a communication between value sending agent and constraint evaluating agent. A two-way communication strategy is imposed. They could interact each other using two types of messages an ok?message and a nogood message.Whenever a value sending agent sends its values to other agent then it send the value as an ok?message asking the constraint evaluating agents that whether its assigned value is consistent with the constraint. All the values received by a constraint evaluating agent by other value sending agents constitute its agent view. The constraint evaluating agent, upon finding any inconsistency in evaluation of constraint with its own possible assignments with its agent view, it will send a nogood messege to one of the values sending agent. A nogood message is a subset of the agent view for which constraint evaluating agent is unable to find any possible consistent value. After sending a nogood message to a value sending agent, it backtracks and changes its value if needed in accordance with its agent view, except the value of the agent to which it sends a nogood message.

Here in the above process, the question arises that how we choose as constraint evaluating agents and how we choose an agent to whom nogood message is sent among value sending agents. To answer the first question, we need to establish a topology of agents connection with incoming (ok?message) and outgoing(nogood message) based on some priority like alphabetical order etc. To answer the second question, the same priority order can be used to send a nogood message to least priority agents among value sending agents.

Since agents changes its values concurrently, it is to be noted that a loop may be formed where one agent change its value simultaneously other agents too changes their values so it may happen that the process does not end. To avoid loops we should use a total order relation among the autonomous agents based on some priority setup. A detailed formalization of DCSP and its initial solution approaches was carried out by yokoo [10] in its paper titled" The distributed constraint satisfaction problem: formalization and algorithms". In his paper, Yokoo provides explanation to three solution techniques to solve a DCSP namely distributed breakout algorithm, asynchronous backtracking algorithm and asynchronous weak-commitment search algorithm. Another approach, asynchronous partial overlay algorithm, to solve a DCSP was given by Roger Mailler [11].

4. Mapping the Problem as a DCSP

First the workload per paper per week is calculated. Before we start scheduling, we check whether the total workload in a semester/year (which is the sum of workloads of each department for the semester/year) is less than or equal to the total timeslots available per week. Here timeslots are the time intervals per class per day. If a class can take 6 lectures in 6 different time intervals in a day and the college runs 6 days in a week then we have 36 different timeslots for that class room. If there are 25 such classrooms available in the college then there are 25x36=900 timeslots available in the college.

Input Formats:

Once we assured that a scheduling process could start, we ask for different types of input for the scheduler. The input is to be stored in formatted tables. In the first table we gather the information about the departments in college/university.

Table 1: Input Table of Departments

Name of Department	Dept-Code
Mathematics	01
Physics	02
Chemistry	03
…	…

Table format for department wise Paper details:
Next, we gather the information about different types of papers that are to be taught in this academic session from each department one by one.
Name of department: Mathematics
Scheme for paper-code:
(Dept − code) – (PaperName) − (GP/NGP) − (T H/PR) − (semester) − (Workload)

Table 2: Paper Details of Mathematics Department

Name of Paper	Genric(GP)/non genric(NGP)	Nature of Paper (Theory/Practical)	Semester	Workload	Paper-code	If GP then involved dept.
MJC01	NGP	TH	1	5	01-MJC01-NGP-TH-1-5	0
MIC01	GP	TH	1	3	01-MIC01-GP-TH-1-3	2,3,6,7
IDC01	GP	TH	1	3	01-IDC01-GP-TH-1-3	2,3,4,5
---	---	---	---	---	---	---

Table format for department wise Faculty details:
Next, we gather the information related to faculty members who are going to be involved in this academic session from each department.
Name of department:
Mathematics
Scheme for faculty-code:
(Dept − code) − (SerialNumber) − (workload)

Table 3: Faculty Details of Mathematics Department

Serial Number	Faculty Name	Designation	Workload	Faculty-Code	Paper-codes that the faculty prefers to teach
1	Faculty 1	Professor	14	01-01-14	01-MJC01-NGP-TH-1-5, 01-MIC01-NGP-TH-1-3
2	Faculty 2	Assistant Professor	16	01-02-16	---
---	---	---	---	---	---

Table format for Classroom details:
Usually there are some class room specifically used by a particular department like physics lecture theatre is used by physics departments only, labarotaries are only be used to held lab class of particular department. we gather this information in the following table format:

Table 4: Classroom Details

Classroom Name	Classroom-Code	LAB(Y/N)	Dept-code if room is specific to the

			department otherwise 0
N1	01	N	00
Phy Lab	02	Y	02
Phy Lecture theater	03	N	02
---	---	---	---

Apart from these input tables, timeslot-code for classrooms is created internally. For example, timeslot-code for classroom-code 01 is as under:
Scheme for timeslot-code:
(classroom−code) − (timeSlotNumber) − (daynumber)

Table 5: Timeslot-Codes

Class room code (let it be 01)	Mon	Tue	Wed	Thu	Fri	Sat
Slot1	01-01-01	01-01-02	01-01-03	01-01-04	01-01-05	01-01-06
Slot2	01-02-01	01-02-02	01-02-03	01-02-04	01-02-05	01-02-06
Slot3	01-03-01	01-03-02	01-03-03	01-03-04	01-03-05	01-03-06
Slot4	01-04-01	01-04-02	01-04-03	01-04-04	01-04-05	01-04-06
Slot5	01-05-01	01-05-02	01-05-03	01-05-04	01-05-05	01-05-06
Slot6	01-06-01	01-06-02	01-06-03	01-06-04	01-06-05	01-06-06

Variables: Three types of variables are present, namely Paper, timeslots and faculty. We create objects of these variables. The objects have the following specifications:
Paper: Paper-object is named with its code and contains the following information:

Table 6: Paper-object blueprint

String timeslot-code [workload]	Store array of timeslot-code
String faculty-code [workload]	Store array of faculty-code
Integer workload	Store workload of the paper
Set and get functions for timeslot-code and faculty-code	
Allotted status	Yes or No, initialized with no

The number of paper object is equal to the total number of papers to be taught in the academic session.
Timeslot: Timeslot-object is named with its code and contains the following information:

Table 7: Timeslot – object blueprint

String paper-code	Store paper-codes
Set and get functions for paper-code	
Allotted status	Yes or no, initialized with no.

The number of timeslot-object is equal to the total number of available timeslots.
Faculty: Faculty-object is named with its code and contains the following information:

Table 8: Faculty-object blueprint

String timeslot-code [workload]	Store array of timeslot-code
String paper-code [workload]	Store array of paper-code
Set and get functions for timeslot-code and paper-code	
Integer Allotted status	0 to workload, initialized with 0 indicating not allotted, 1 to workload, indicates the number of lectures allotted to the faculty.

5. Scheduling Strategy

The scheduling work is done at two levels. At the first level we only assign timeslots-codes to paper objects. This we term as Master timetable

schedular (MTTS). At the second level, we assign faculty-codes to paper objects. We named this level as Department timetable scheduler (DTTS).

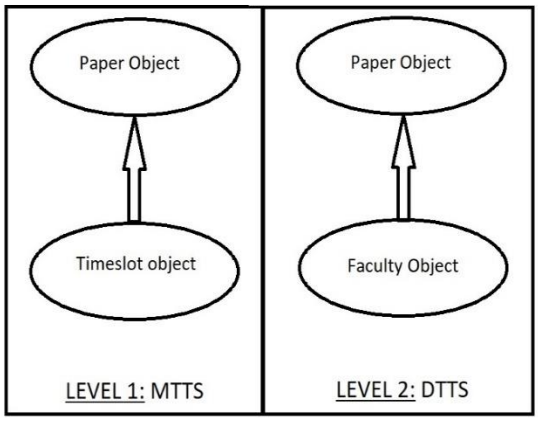

Figure 2: Two level of Scheduling

MTTS:

Here, departments are considered as autonomous agents. Number of agents = number of departments. At this level of scheduling, each department start scheduling their respective generic papers from the table of operations. Once all the generic papers of all departments are scheduled then non generic papers are scheduled. The following constraints are considered here:

Constraint 1:

Assignment is to be done up to the workload of the paper.

Constraint 2:

More than one lecture of the same papers if assigned in a day then those lectures must be consecutive.

Constraint 3:

If D1, D2, …, Dr be the departments involved in a generic paper then at the assigned timeslots to generic paper, there should not any assignment to the particular semester students of D1, D2, , Dr . For example, if a timeslot 01-01-01 is assigned to a generic paper say 01-MIC01-GP-TH-1-3 and dept-codes 02(physics), 03(chemistry), 06(computer science), 07(economics) are involved in this generic paper then the timeslot-code that ends with 01-01 that is *-01-01 should not be assigned to semester 01 of the above involved departments.

Topology of MTTS agents:

A topology of agents is being formed where two agents are interconnected if a generic paper is involved between these two agents (Departments). Direction of interconnection is based on dept-codes.

Note:

1. There are two sets of timeslot codes are to be maintained namely allotted group and not-allotted group. Initially all the timeslot-codes are not allotted. Assignment is to made from not allotted group.
2. Some classrooms are department specific. So, first we these classroom timeslots are to be used by respective departments. If the classroom is a lab then it can only holds Lab class.

Upon the successful completion of MTTS, we are certain to obtain a feasible schedule as the scarcest resource (timeslots) has been utilized. The output of MTTS is a schedule that consists of paper objects with assigned timeslots. The added advantage of this approach is that we need not to share paper details to respective departments.

DTTS:

At this level of scheduling, Paper objects are considered as autonomous agents and faculty-codes are to be assigned to these paper-object agents. Each department runs DTTS concurrently as no two department has same faculty. The following constraints needed to be checked:

Constraint 1:

Assignment of faculty-code to paper-code objects to be done on the basis of preference table consists of paper-codes and corresponding faculty-codes of those faculty who likes to teach that paper.

Constraint 2:

If we assign a faculty-code to a paper-code object then we actually assign associated timeslot of paper-code to faculty-code. So, assigning faculty-code to a paper-code is nothing but to assign timeslot-code to faculty-code. Now, if two timeslot-code say a-b-c and x-y-z are assigned to a faculty code then b must not be equal to y and c must not be equal to z. In this way, a faculty cannot be assigned to two different classrooms at the same timeslot.

Constraint 3:

Assignment of timeslots to faculty is done up to its workload.

Topology of DTTS agents: Two agents (Papers) are interconnected if the intersection of faculty preferences of these papers is non empty. The direction of interconnection is based on the serial number of paper-codes in preference table.

Note:

There are two sets of faculty-codes are to be maintained namely CompletelyAllotted group and Not-CompletlyAllotted group. Initially all the faculty-codes are in the second group. Assignment is to made from second group.

Once DTTS runs successfully, we obtain a schedule where each paper-code object is assigned with timeslotcode and faculty-code, each faculty-code is assigned with paper-code and timeslot-code.

6. Conclusion

Since the scheduling is done at two levels, constraints are partitioned into two non-overlapping groups. Once the MTTS get a schedule satisfying first set of constraints, then we are certain to get a final schedule by relaxing some constraints at second group or adding suitable faculty resource for DTTS.

Usually, scheduler based on CSP requires a lot of input from the user. This DCSP based scheduler is designed to minimize the input from the user. It also helps to minimize constraint checking. The DCSP scheduler reduces the search time as the assignments to variables are done concurrently.

One of the major advantages of this scheduling design is that it can easily be implemented by any object-oriented programming language. The design is capable enough to be used by any colleges/universities in India with minor modifications.

In future, some heuristics like local search, Hill climbing or Ant colony optimization can be incorporated to further reduce the search time for a feasible solution.

7. Competing Interests

No competing interests.

References

[1] M. W. Carter and G. Laporte, “Recent developments in practical course timetabling,” in Practice and Theory of Automated Timetabling II, Lecture Notes in Computer Science, Vol. 1408, edited by E. K. Burke and M. W. Carter (Springer Berlin Heidelberg, 1998), pp. 3–19.

[2] C. Valouxis and E. Housos, Computers & Operations Research 30, 1555–157209 (2003).

[3] S. N. Jat and S. Yang, Journal of Scheduling 14, 617–637 (2011).

[4] H. Andersson, School timetabling in theory and practice a comparative study of simulated annealing and tabu search, 2015.

[5] R. A. Oude Vrielink, E. Jansen, E. W. Hans, and J. van Hillegersberg, Annals of operations research 275, 145–160 (2019).

[6] A. Golabpour, H. M. Shirazi, A. Farahi, A. Z. M. Kootiani, and H. Beigi, “A fuzzy solution based on memetic algorithms for timetabling,” in 2008 International Conference on MultiMedia and Information Technology (IEEE, 2008), pp. 108–110.

[7] J. A. Soria-Alcaraz, G. Ochoa, J. Swan, M. Carpio, H. Puga, and E. K. Burke, European Journal of Operational Research 238, 77–86 (2014).

[8] T. Birbas, S. Daskalaki, and E. Housos, Journal of Scheduling 12, 177–197 (2009).

[9] G. Solotorevsky and E. Gudes, “Algorithms for solving distributed constraint satisfaction problems (dcsps),” in Artificial Intelligence Planning Systems Conference (AIP), edited by B. Drabble (AAAI Press, Edinburgh, 1996), pp. 191–198.

[10] M. Yokoo, E. Durfee, T. Ishida, and K. Kuwabara, IEEE Transactions on Knowledge and Data Engineering 10, p. 673685 (1998).

[11] R. Mailler and V. Lesser, “Using cooperative mediation to solve distributed constraint satisfaction problems,” in Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. (IEEE, 2004), pp. 446–453.