Journal of Computer Science

Original Research Paper

Exploring a Flexible Scoring Scheme for a Heuristic Search Technique: A Case Study of University Timetables

[1]Sangsuree Vasupongayya, [1]Warakorn Sitthirit, [1]Suthon Sae-Wong and [2]Thaniya Kaosol

[1]Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, Songkhla, Thailand
[2]Department of Civil Engineering, Faculty of Engineering, Prince of Songkla University, Songkhla, Thailand

Article history
Received: 13-10-2019
Revised: 07-03-2020
Accepted: 28-05-2020

Corresponding Author:
Sangsuree Vasupongayya
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, Songkhla, Thailand
Email: vsangsur@coe.psu.ac.th

Abstract: The university course timetabling management system is proposed in this work. The proposed system applies depth-bounded discrepancy search together with the heuristic and the hard-soft constraints to guide the search to a good solution space. To make the proposed system easy to use by a non-technical user, this work explores a scoring scheme including the effects of the ordering heuristics, the effects of the objective model and the effects of the workload characteristics in order to pre-define a set of configurations and to automatically adapt to the changes in the workload. The experimental results show that the proposed system can find a suitable solution for various workloads. The feedback from the potential users on the proposed system is positive.

Keywords: Depth-Bounded Discrepancy Search, Constraints, Objective Model, Timetable

Introduction

A university course timetabling management system is proposed in this work. A university course timetabling problem can be classified as a resource allocation problem (Murray *et al*., 2006). The requirements in each institute may be slightly different depending on the resources and the specific conditions (Chaiyasuwan and Tantithamphusit, 2007a; 2007b). For example, a timeslot to be assigned to a course must avoid the weekly faculty meeting; some institutes allow evening classes. Thus, the constraints must be adapted to fit the institute requirements. Moreover, the requirement can be defined by many agents such as the department or the curriculum (Chaiyasuwan and Tantithamphusit, 2007c; Ameen, 2012).

At Prince of Songkla University, Thailand, which is the case study institute in this work, the courses offered by various programs have different characteristics. For example, the courses offered by Faculty of Science, Faculty of Engineering and Faculty of Agro-Industry are typically 1-h-morning classes. While, the courses offered by Faculty of Management Sciences and Faculty of Economics are typically 1.5-h-afternoon classes (Chaiyasuwan and Tantithamphusit, 2007c). Many curriculums contain a lot of laboratory-style courses while some curriculums contain a lot of small-size or large-size lecture-style courses. Each curriculum can also place some restrictions on the room/equipment usages. To add more complexity to the problem, some restrictions can be changed for each semester. Therefore, a course timetabling management system that can adapt to the requirements, constraints and changes is required.

This work aims to explore a method to describe a set of requirements in terms of heuristics and constraints that the users are familiar with, in order to be used as a guide for the semi-automatic timetable management system to find a suitable timetable for each curriculum even when a set of requirements have been changed. The Depth-bounded Discrepancy Search (DDS) is applied as the main search algorithm of the proposed system while the heuristic and the hard-soft constraints is applied to guide the search to a good solution space. To adapt to the workload characteristics, the proposed system allows the constraints to be configurable. The results in this study are used as the pre-defined system configuration. Even though this work is focusing on a single institution as the case study, the approach can be applied to other similar institutions.

Literature Review

In this section, the related works and techniques to be used in this work are presented, including a literature review on the solution to the university timetabling problems, the constraints types, the hard/soft constraints and the objective models. Lastly, the depth-bounded discrepancy search algorithm is explained.

University Timetabling

A university course timetabling management system can be classified into two groups (Bonutti *et al.*, 2012). The first group is called a post-enrolment based course timetabling, which is done after the student enrolment period. The second group is called a Curriculum-Based Course Timetabling (CB-CTT), which is done before the student enrolment period. The case study in this work belongs to the second group.

Under the CB-CTT group, four input information including the course information, the timeslot information, the room information and the curriculum information are required. First, the course information includes a set of sections, a set of teachers, a set of courses, the room type for each courses, the student capacity and the student groups. Second, the timeslot is a set of available date and timeslots for scheduling the courses. Third, the room information includes the type of rooms, the number of rooms in each type and the room capacity. The curriculum information for each student group describes the offered courses.

Currently, many researchers are still interested in the course timetabling problem. Mathematics models are applied in several methods including integer linear programming methods (Lach and Lübbecke, 2012; Sánchez-Partida, 2013; Fink Bagger *et al.*, 2019; Boland *et al.*, 2008; Antony *et al.*, 2017) and reduced graph coloring methods (Burke *et al.*, 2010). Logic programming methods (Banbara *et al.*, 2013) are also applied. The metaheuristic approaches, which can be categorized into two groups, are used for solving the course timetabling problems (Lewis, 2008). The first metaheuristic method group is called a single solution metaheuristic method, including simulated annealing methods (Bellio *et al.*, 2016a; 2016b; Gunawan *et al.*, 2012), iterative local search methods (Soria-Alcaraz *et al.*, 2016; Goh *et al.*, 2017; Song *et al.*, 2018) and Tabu search methods (Wangthammang *et al.*, 2018; Lü and Hao, 2010). Another metaheuristic method group is called a population-based metaheuristic method including genetic algorithm methods (Badoni *et al.*, 2014; Jain *et al.*, 2010; Chinnasri and Sureerattanan, 2010), ant colony optimization algorithm methods (Nothegger *et al.*, 2012), artificial bee colony algorithm methods (Bolaji *et al.*, 2014), harmony search algorithm methods (Al-Betar and Khader, 2010), swarm intelligent methods (Turabieh *et al.*, 2010) and honey bee mating optimization algorithm methods (Sabar *et al.*, 2012).

Some approaches can be combined as a hybrid method for improving the performance and reducing the disadvantages of the single algorithms. Hybrid methods include a combination of mathematic model with metaheuristic methods (Gunawan *et al.*, 2012) and a single solution metaheuristic method with a population-based metaheuristic method (Lü and Hao, 2010; Bolaji *et al.*, 2014; Kohshori and Abadeh, 2012) and a single solution with metaheuristic methods (Bellio *et al.*, 2012).

However, the performance of these algorithms depends greatly on the characteristics of the dataset. The third International Timetabling Competition (ITC2011) (Post *et al.*, 2016) shows that no single method can dominate the others for all types of datasets. Some methods are better than another but they are also worse on some datasets. Thus, it is impossible to create a method that suits all datasets, because it depends on the institute rules, features, costs and fixations (Bonutti *et al.*, 2012). To bridge the gap between theory and practice, several works focus on real-world implementations (Müller and Rudová, 2016; 2014). Some works create a guideline to measure or to benchmark the results (Bonutti *et al.*, 2012; Schaerf and Di Gaspero, 2007). McCollum (2007) suggests that the interface of a practical system must be assistive such that the user can model the dynamic constrains easily.

Constraint Types

The course timetable must be correct according to the requirements. The requirements can be viewed as a set of constraints. Thus, the constraints can be used for guiding the search to a suitable solution. The timetabling constraints can be categorized into five groups (Lewis, 2008) including unary, binary, capacity, event-spread and agent. To be practical, the structure of each timetabling constraint group will be considered in designing the proposed system.

First, the unary constraint considers only one course or one group of courses. The unary constraint requires three inputs to create a relation between the course and the rooms. The relation puts restrictions or permissions on how to use the rooms.

Second, the binary constraint considers the relation between two courses. For example, a drawing lecture-style course must be scheduled before a drawing laboratory-style course. A binary constraint also requires three inputs. However, the relation between two courses is more complex.

Third, a capacity constraint considers the capacity of the room. That is, the room must have enough seats to hold the course. The capacity constraint requires two inputs including the course and the number of seats.

Forth, an event spread constraint considers the space between courses. The event spread constraint is either spreading the courses out or clumping the courses together, depending on the user requirements. The event spread constraints can be controlled by the period of free timeslots between courses and the number of maximum total hours per day. If the users require a spreading-out timetable, they should either increase the maximum number of free timeslots between courses or decrease the maximum total hours per day. On the other hand, if the users require a clumping-together timetable, they should either decrease the maximum number of free timeslots between courses or increase the maximum total hours per day. An event spread constraint can be defined by two information including the maximum number of free timeslots between courses or the maximum total hours per day.

Fifth, an agent constraint considers the party in each timetable including the teachers and the student groups. The agent constraint puts a specific condition on the problem such as the relation between the students and the rooms. The agent constraint requires three inputs including the agents, the relations and the resources where the relation will be given as a unary constraint on the resource.

Hard/Soft Constraints

As mentioned above that the constraints can be used for guiding the search in order to find a suitable solution. The type of the constraints can affect the result. In addition, some constraints are more important than others. For example, the constraint of assigning available rooms and timeslots for all courses according to the requirements is more important than the constraint of assigning the afternoon timeslots for the laboratory-style courses. Hence, some laboratory-style courses can be scheduled in the morning timeslots but all courses must be assigned the rooms and timeslots.

The violation of some constraints can cause problems. For example, if the constraint of assigning the available rooms for a course is violated, then two courses might be assigned to the same room. Thus, there are some constraints that cannot be violated which is called hard constraints. The other type of constraints are called soft constraints.

Objective Models

During the search in the solution space, the proposed system must have an ability to evaluate the solution found in order to find a suitable solution. To evaluate n intermediate solution, an objective model is used. The objective model is a method to compare two solutions in order to select a suitable solution among solutions in the search space. To compare two solutions, a scoring technique is required to determine which solution is better. The number of the constraint violations can represent the quality of a solution. The less constraint violation solution is better than the more constraint violation one. However, each constraint is different. The hard constraints must dominate the soft constraints. Some soft constraints might be preferred over the others. Thus, the system requires a model to determine how to compare the constraints. The objective model can help solving this issue.

This work applies multi-objective model (Vasupongayya and Chiang, 2006), including lexical, order-tradeoff and equal-tradeoff models. Lexical model (denoted Lexical (A->B), where A and B are constraints). The model defines that A is more important than B. Thus, A will dominate B in all cases. Next, ordered-tradeoff model (denoted Tradeoff (A->B), where A and B are constraints). The model defines that A is more important than B. However, if the improvement of B is larger than the degradation of A, B which is less important can also win the competition. Last, equal-tradeoff model (denoted Tradeoff (A: B), where A and B are constraints). The model defines that A and B are equal. Thus, the improvement and degradation of both constraints can affect the result equally.

Depth-Bounded Discrepancy Search

Depth-bounded Discrepancy Search (DDS) (Walsh, 1997) is a complete search technique. DDS is used in this work because the main goal of this work is to allow the users to understand and to configure the system. By applying the search algorithm with constraints, the proposed system allows the non-technical users to configure the system easily. The DDS search space can be viewed as a search tree. For explanation purpose, Fig. 1 shows the search tree of eight solutions.

The search will start from the root node to the leaf node. Each solution is a path from the root node to the leaf node. This way, once the search reaches the leaf node then a candidate solution is found. At this point, the objective model will be used for calculating the quality of the solution in terms of a score. If the new solution is better than the current best solution, then the new solution is recorded as the current best solution. The search continues and the score will be calculated every time that the search reaches the leaf node. If the solution is no better than the current best solution then the path can be skipped in order to accelerate the search process. This effect is called pruning.

The progress of DDS will also be affected by the ordering of the branch at each level. That is, the search will start from the left-most path (A, B, D, H) which is called the heuristic path. The heuristic is a guide of how to order the node at each level. For example, if each node is the course to be scheduled. The order of the courses to be considered by the scheduler is the heuristic.

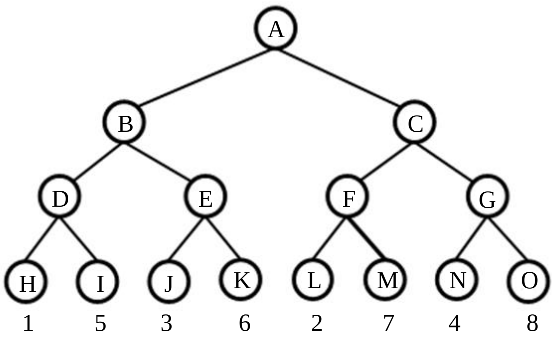

Fig. 1: Depth-bounded Discrepancy Search (DDS)

The heuristic might be wrong in some cases and the effects of such incorrect heuristic can be monstrous. DDS prevents such gigantic effects by probing into each subtree in order to get a good baseline solution for pruning the search space. For example, Fig. 1 shows the order of the leaf nodes to be discovered by the DDS. The first leaf is the left-most path (A, B, D, H). The next path is the 2th path (A, C, F, L) which is a path from a different subtree. Thus, the first path visited by the search engine is from the left-most subtree while the second path visited by the search engine is from the right-most subtree. This property is suitable for the course timetabling problem because the earlier course assignments can have a huge impact on the later courses. Furthermore, the objective model can be calculated on a partial solution for the lexical model. Thus, the pruning technique can be used to avoid searching in the area containing solutions that are not better than the current solution found so far.

The results on hard/soft constraints using DDS as the main search engine was presented in our previous work (Sitthirit and Vasupongayya, 2013). However, the results only include the ability of the search engine to reduce the number of hard constraint violations while improving the number of soft constraint violations. Moreover, the set of hard and soft constraints are slightly different in this work. This work provides more detail performance analysis on the proposed system than that shown in (Sitthirit and Vasupongayya, 2013).

Proposed System

This section provides the system overview and the scoring technique used in our proposed system.

System Overview

The course timetabling system requires the course data including courses, teachers and curriculums as the resources. The heuristics specify the dataset to be chosen to create a search tree. The constraints are used as a decision to evaluate the results. The output of the system is the set of courses with rooms and timeslots that are assigned by the search engine. The input and output of the system are shown in Fig. 2.

The system uses Depth-bounded Discrepancy Search (DDS) as the main search engine to assign the rooms and timeslots for each course. The search engine first creates the search tree by using the timetabling and heuristic data. Each node of the search tree represents a course. Each path of the search tree represents a solution. This way, the length of all paths is equal. Thus, the whole search tree contains all possible solutions. The heuristics which are defined by the scheduler will order the nodes in each path, where the left-most path is the heuristic path. The search engine is then search for a suitable solution by evaluating each solution found so far using the scoring technique.

The size of the search tree depends on the dataset. The scheduler can choose all courses in each semester to create the search tree or choose a partial data for each search tree. However, the length of each path determines the length of the processing time for each solution, because the larger search tree will result in the longer search time. The necessary courses that should be in the same search tree are the courses that have a relation among themselves such as the courses with the mutual student groups or the courses with the mutual teachers. In this work, the search trees are created according to the student groups, because these students have mutual courses.

The order of nodes in each path is defined by the ordering heuristic. Figure 3 shows an example of an ordering heuristic which can be used for creating the search tree. The student group will be decided first. For each student group, the laboratory-style courses will be considered before the main, elective and free elective courses, respectively. For the same course type, the course hours will be used for ordering the courses. For example, a four-hour laboratory-style course will be considered before a three-hour laboratory-style course.

To allocate the rooms and timeslots, the search engine builds the scoring table of all available timeslots in each node of the search tree. The scoring engine has to check the availability of all related teachers, rooms and student group timeslots. Next, the scoring engine will calculate the constraint violations for all available timeslots and return the score back to the search engine to create a scoring table. The search engine will compare all scores in the scoring table to choose the best timeslot and room. Next, the search goes along the path to get a complete solution and compares the score of the current solution and the score of the best solution found so far. The scoring technique is described next.

Scoring Technique

The search engine requires the scoring technique in order to determine the score of the discovered solution during the search process. The simple method to calculate the score of the current solution is counting the number of constraint violations. However, each timetabling constraints are of different types. Thus, the search engine requires a scheme to count the number of violations for each constraint type. The unary and binary constraints are counted by the number of violations that are occurred and multiplied by the number of student group timetables related to the course, to weight in the size of its effect. The violation of the time-spread constraints is the number of timeslots.

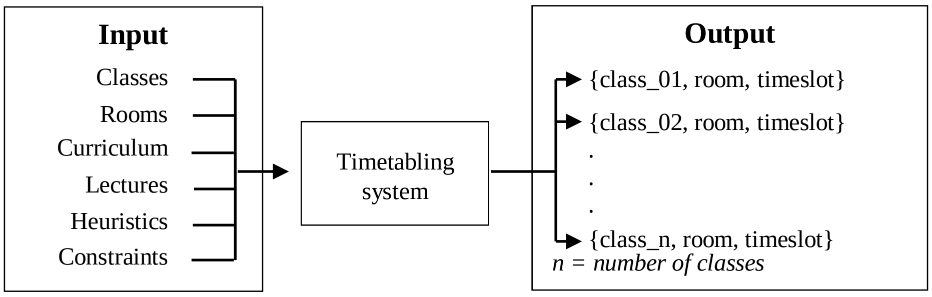

Fig. 2: Input and output of the proposed system

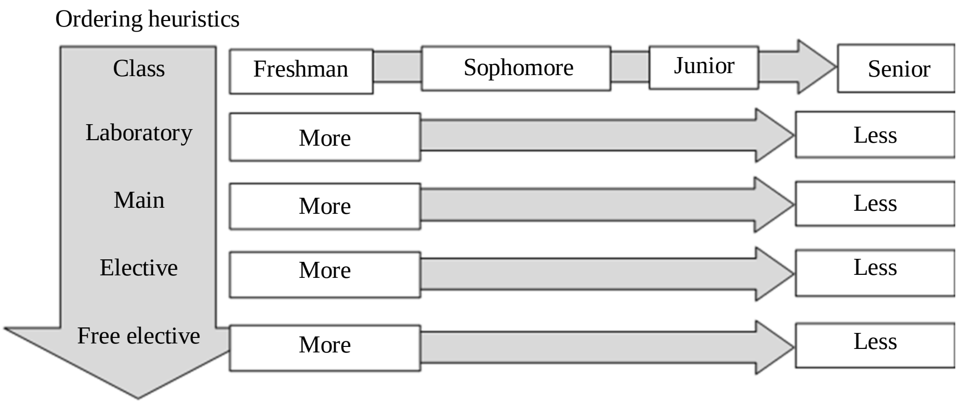

Fig. 3: Ordering heuristic for creating the search tree

The scoring algorithm process starts from creating a scoring table. The hard constraints are the constraints that should not be violated. If any hard constraint is violated, the result will be discarded. The comparison between two Hard Constraints (HC1, HC2) uses equal-tradeoff (denoted Tradeoff (HC1: HC2)) because the importance of all hard constraints are equal. However, some hard constraints are allowed to be violated. The scoring engine will count all allowed hard constraint violations before the score will be stored in the scoring table.

As hard constraints dominate all soft constraints, the comparison between a Hard Constraint (HC) and a Soft Constraint (SC) is a lexical model (denoted Lexical (HC-> SC)). This way, any soft constraint violation cannot overrule the effect of any hard constraint violation.

The comparison among different Soft Constraints (SC1, SC2) is ordered-tradeoff model (denoted Tradeoff (SC1-> SC2)) meaning constraint SC1 is more important than constraint SC2. In the case that the decreasing of SC2 violations is more than the increasing of SC1 violations, the SC2 which is less important can be preferred over SC1. The comparison among equal important Soft Constraints (SC3, SC4) is equal-tradeoff model (denoted Tradeoff (SC3: SC4)).

Experimental Settings

This section describes the experimental settings including workloads, workload characteristics, modified workloads and the experimental details.

Workloads

In this work, the workload is a set of courses offered by the case study institution. There are 674 courses, 57 rooms and 66 student groups from 12 curriculums of 7 departments including Mechanical Engineering (ME), Civil Engineering (CE), Chemical Engineering (ChE), Electrical Engineering (EE), Industrial Engineering (IE), Mining Engineering (MnE) and Computer Engineering (CoE). However, the first year students are separated as a special group. The first year students have the largest number of hours following by the mechanical engineering student group and the computer engineering student group.

To maintain a small tree size and allow the changes to parts of the problem, the tree is created according to the student groups. The freshman students are divided into two groups according to the current university setting. The sophomore, junior and senior students are divided into 7 groups according to their departments. This way, the students with shared classes are grouped together. The order of consideration within the same level is the total course hours. As a result there are now 23 datasets.

The datasets consist of many different courses. Each dataset has its own characteristic. For example, the freshman timetables contain a lot of study hours but most of them are pre-defined. 2ME timetable has similar numbers of 1-, 2- and 3-h courses, while 3IE timetable contains a large number of 3-hour courses. The number of teachers for each course is also considered. Some datasets have less multiple teachers while some datasets have many multiple teachers. Moreover, the total hours of each dataset and the room requirement are also considered. With variety of workload characteristics, the results in this work can be generalized to other institutes.

Workload Characteristics

The following list of rules shows the characteristic of the datasets.

Different-class-size: If any group of courses in the dataset has a high Standard Deviation (SD) of course size, then the group is considered to have a different-class-size characteristic. A different-class-size characteristic requires a different space size in the schedule. As a result, finding the exact space size for each course will be difficult.

Multiple-teacher: If any group of courses in the dataset contains a large number of courses that are taught by many teachers then the group is considered to have a multiple-teacher characteristic. This characteristic will add a constraint on the available space of the courses.

High-workload-teacher: If any group of courses in the dataset contains a large number of courses that are taught by high load teachers then the group is considered to have a high-workload-teacher characteristic. The high load factor can be calculated by the summation of the square value of each period of each course in a week. That is, a large slot will result in a higher load value because a large slot is more difficult to find than a smaller one. This characteristic will add a constraint on the available slots of the courses.

Teacher-time-conflict: If any group of courses in the dataset allows the teacher-time conflict result then the group is considered to have a teacher-time-conflict characteristic. This characteristic will relax the constraints.

More-total-hour: If any group of courses in the dataset has a total hour larger than the median value of all datasets then the group is considered to have a more-total-hour characteristic. This characteristic will affect the available slots for the other courses.

Multiple-student-group: If any group of courses in the dataset contains a large number of multiple student groups then the group is considered to have a multiple-student-group characteristic. This will affect the available slots in the schedule.

Elective-subject: If any group of courses in the dataset contains an elective course then the group is considered to have an elective-subject characteristic. For some institute the elective courses must be flexible for all students. As a result the elective courses will relax the schedule. However, some elective courses have a high priority then it will add more constraints to the problem.

Multiple-room: If any group of courses in the dataset contains a large number of courses with multiple rooms then the group is considered to have a multiple-room characteristic. This will add constraints on the room requirements.

The above rules are applied on the datasets in order to classify their characteristics into 9 groups as follows. The first characteristic is called “different-laboratory-course-size” (denoted CH_1) and there are 8 datasets with this characteristic. The second characteristic is called “different-lecture-class-size” (denoted CH_2) and there are 14 datasets with this characteristic. The third characteristic is called “multiple-teachers” (denoted CH_3) and there are 12 datasets with this characteristic. The forth characteristic is called “high-workload-teacher” (denoted CH_4) and there are 12 datasets with this characteristic. The fifth characteristic is called “teacher-time-conflict” (denoted CH_5) and there are 14 datasets with this characteristic. The sixth characteristic is called “total-hours” (denoted CH_6) and there are 13 datasets with this characteristic. The seventh characteristic is called “multiple-student-groups” (denoted CH_7) and there are 7 datasets with this characteristic. The eighth characteristic is called “elective-subject” (denoted CH_8) and there are 8 datasets with this characteristic. The ninth characteristic is called “multiple-room” (denoted CH_9) and there is one dataset with this characteristic.

Numbers of characteristics of a single dataset can be used as an indicator of the difficulty level to allocate the rooms and timeslots. For example, 3CE dataset which contains 6 characteristics has a high possibility to cause a problem during the scheduling process than 2EE dataset which contains only 2 characteristics. The experimental setting in this work aims to evaluate the effects of constraints and ordering heuristics on various dataset characteristics in order to define the search engine rules.

Modified Workloads

To create the different dataset characteristics, the original data set (UAs) are modified into several different characteristic datasets as a set of Testing Datasets (TD). There are 4 modified datasets including TD1, TD2, TD3 and TD4. Under TD1 workload, the original dataset is modified by splitting all 2-h, 3-h courses to one-hour courses. Thus, the TD1 dataset contains a lot of short-class types. For the TD2 workload, the original dataset is modified by splitting all courses to 1.5-h courses. For the TD3 workload, the original dataset is modified by splitting or combining all courses to 2-h courses. For the TD4 workload, the original dataset is modified by combining all courses to 3-h courses.

Thus, the experiments are conducted on five datasets (UA, TD1, TD2, TD3 and TD4) and 6 different Ordering Heuristics (OH1, OH2, OH3, OH4, OH5 and OH6). Totally, there are 30 sets of results.

Experimental Details

Four performance results will be studied in this work. First, the performance of the original timetables is presented as the baseline of this work. The next three experiments are conducted on the proposed system. Second, the effects of the search time are studied. Third, the effects of the objective models are studied. Forth, the effects of the ordering heuristics are studied.

For comparison purpose, the details of the original university timetables are presented first. The case study institute offers 12 undergraduate degrees in engineering fields. There are total of 143 teachers, 674 classes from 202 courses and 57 rooms. Each course can consist of several classes. The courses include 93 laboratory-style classes and 581 lecture-style classes. The teachers are grouped into 7 departments. The rooms are located in different buildings. Thus, the scheduler has to allocate rooms and timeslots for a lot of classes in each semester.

To study the effects of the search time, both the actual runtime of the proposed system and the number of iterations executed by the proposed system are studied in order to show how fast the proposed system can find a suitable solution.

Three objective models are conducted in this work. Three objective models include (1) Simple Mixed Objective (SMO) that assigns an equal priority to all soft constraints, (2) Mixed Objective 1 (MO1) that assigns a high priority to Time Constraints (TC) and (3) Mixed Objective 2 (MO2) that assigns a high priority to Time-Spread Constraints (TSC). The constraints to evaluate the results are grouped into 3 sets including hard constraints, time constraints and time-spread constrains.

There are six ordering heuristics as shown in Table 1. To study the effects of the ordering heuristics on different dataset characteristics, six orders of three workload characteristics are created. The three workload characteristics include the class size, the teacher workload and the study-hour.

Table 1: Ordering heuristics

Index	Order
OH1	Class size > Teacher > Study-hour
OH2	Class size > Study-hour > Teacher
OH3	Teacher > Class size > Study-hour
OH4	Teacher > Study-hour > Class size
OH5	Study-hour > Class size > Teacher
OH6	Study-hour > Teacher > Class size

To achieve the practical points of the proposed system, the usability side of the proposed system is also evaluated and discussed. The discussion include the guide for users and the user interface. The proposed system is also evaluated by potential users.

The DDS will create a set of search trees according to the ordering heuristic. The constraints are applied to the search engine with an equal priority of soft constraints. HC is used for representing the number of hard constraint violations and SC is used for representing the number of soft constraint violations.

Results and Discussion

Results on original time tables are given first as the baseline of this work. The next three section provides the performance of the proposed system on the effects of the search time, the effects of the objective models and the effects of the ordering heuristics. Next, the guide for the users and the user interface of the proposed system are presented. Last, the proposed system is evaluated by potential users.

Original Timetables

In summary, the original university timetable produces 820 soft constraint violations and 399 hard constraint violations. The soft constraint violations include 797 time-constraint violations and 23 time-spread-constraint violations. Even though the workload is not extremely large, the issues and conflicts are still observable. According to the violations of the original university timetables, the workload characteristics can increase the complexity to the process. For example, the electrical laboratory courses are required in many curriculums. Such courses link many student groups; such courses require multiple teachers; and such courses add an extra complexity to the problem.

Effects of the Search Time

The first set of results focuses on evaluating the performance of the proposed solution on the original dataset in comparison with the original university timetables. Two sets of results produced by the proposed system with a short runtime (denoted first depth) and a longer runtime (denoted second depth) are provided in order to display the ability of the proposed system to find a better solution when more time is given.

Table 2 provides the number of constraint violations of the original university timetables and that of the timetables produced by the proposed system. The experimental results show that the proposed system reduces the number of violations in comparison with that of the original university timetables. The total number of hard constraint violations is reduced from 399 to 100 and 90 for the first depth DDS and the second depth DDS, respectively. The total number of soft constraint violations is also reduced from 820 to 533 and 506 for the first depth DDS and the second depth DDS, respectively. The results confirm that the core mechanism of the proposed system is working well in finding a better solution when more time is given.

The proposed system is evaluated on a machine with Intel(R) Core(TM) i5-2400 3.10 GHz Central Processing Unit (CPU) and 8 Gigabyte memory. The proposed system takes approximately 0.462 seconds to process each node. The processing time can be reduced by adding the data catching technique into the proposed system.

Effects of the Objective Models

This section evaluates the effect of three objective models. Table 3 shows the soft constraint violations of all three objective models while Table 4 shows the hard and soft constraint violations of all three objective models. At the first look, the hard constraint violations of all three objective models from the proposed system are reduced from that of the original university timetables, presented in Table 2.

The freshman timetables are better when MO2 is applied. Since the freshman timetables have limited available timeslots, the restriction on the Time-Spread Constraints (TSC) is more effective. The sophomore timetables are considered early in the search process. Therefore, there are a lot of available slots. The junior and senior timetables are considered later in the search process, resulting in limited slots. However, MO1 shows a few number of Hard Constraint (HC) violations in comparison with that produced by MO2.

In conclusion, MO1 which emphasizes on the Time Constraint (TC) produces a better solution than that of the MO2 which emphasizes on the Time-Spread Constraints (TSC). Thus, the Time Constraints (TC) have more effects on the proposed system than the Time-Spread Constraints (TSC). However, the SMO results which regards all constraints as equal, show a promising results. Therefore, an equal priority can help in avoiding the effects of constraint types for the datasets that are complex.

Effects of the Ordering Heuristic

To evaluate the effects of the ordering heuristic, six ordering heuristics are applied on the modified workloads. Table 5 shows the experimental results of all six ordering heuristics on all five datasets. The experimental results show that a completed timetable of the original dataset (UA) can be achieved by OH1. The OH1 can provide a completed timetable on most of the modified datasets, in comparison with other ordering heuristics.

Table 2: Constraint violations comparison

	Number of constraint violations			
	HC	TC	TSC	SC
Original	399	797	23	820
First depth	100	513	20	533
Second depth	90	483	23	506

Table 3: Objective model performance comparison

	Number of constraint violations		
Model	TSC	TC	SC
SMO	347	21	368
MO1	366	15	381
MO2	326	29	355

Table 4: Detail comparisons among objective models

	Number of constraint violations					
	Model	1st	2nd	3rd	4th	Total
SMO	HC	0	4	10	34	48
	TSC	26	100	138	83	347
	TC	1	10	5	5	21
MO1	HC	0	4	12	36	52
	TSC	26	122	147	71	366
	TC	1	5	6	3	15
MO2	HC	0	4	10	40	54
	TSC	15	89	147	75	326
	TC	8	11	5	5	29

Table 5: Ordering heuristic performance comparison

	Number of incomplete schedules					
Order	UA	TD1	TD2	TD3	TD4	Total
OH1	-	5	1	-	2	8
OH2	1	5	2	1	2	11
OH3	1	5	3	1	5	15
OH4	1	5	3	1	2	12
OH5	-	5	3	-	3	11
OH6	-	5	2	-	3	10

In conclusion, the performance of all six ordering heuristics on the modified datasets shows that the workloads with a large number of small class-size courses might run into an issue with the Time-spread Soft Constraints (TSC). If the class-size constraint is already considered then the teacher load constraint is a better next-order constraint to be used than the study-hour constraint. TD1 and TD2 datasets contain a lot of courses which can increase the difficulty in the scheduling process. Reducing the number of courses by creating a combination of two-hour and one-hour courses produces a better performance under OH1, OH5 and OH6 because the complete schedules can be achieved.

Guideline for Users

To provide an ease of use for the users with less technical background, many parameters must be pre-configured such that the output will meet the user satisfaction. First, the datasets of unrelated courses must be divided into different trees in order to speed up the searching process. If any curriculum contains the rooms or the teachers that are not overlapped with each other, then such curriculums can be processed in parallel. This way, the processing time can be reduced. Second, the ordering heuristic to be used as a baseline is OH1. If the workload does not contain various class-size then OH6 can also be applied.

User Interface

The user interface of the system must be classified into two sets. The first set is the search engine parameters. For the search engine parameters, several parameters are pre-defined while several parameters are automatically generated after the workload is fed into the system. The pre-defined parameters include a set of hard constraints, a set of soft constraints, a preferred objective model and a preferred ordering heuristic.

The second set is the specific set of constraints. Figure 4 shows an example of the user interface for configuring the constraint. According to Fig. 4, the A box is the agent type to be defined; the B box allows the user to searching for the target of the constraint to be defined; the F box is an example of the permission on the constraint; the G box shows the area for the users to define the allow-period and the non-allow period. The B, C, D and E boxes allows the user to search for the target or the agent type in the target. As can be seen that instead of the text-based input, the user interface is visualized for ease of use. This way, both technical and non-technical users can use the proposed system easily.

Potential User Feedback

After the proposed system has been developed, the proposed system is then evaluated by three types of potential users including the curriculum head, the curriculum secretary and the temporary staff. These three groups of users are selected to evaluate the proposed system in terms of its ease of uses.

The feedback from all three potential users on the proposed system is positive. That is, the proposed system can always produce the workable timetable from the curriculum secretary perspective; various output formats including the schedule for each student group, the schedule for each teacher and the schedule for each room are beneficial to the curriculum head to manage the curriculum or to organize an extra-curricular activity. The temporary staff can also configure the proposed system in order to find the timetable for her curriculum. However, the temporary staff takes a longer time to configure the system because she is unfamiliar with the timetabling task.

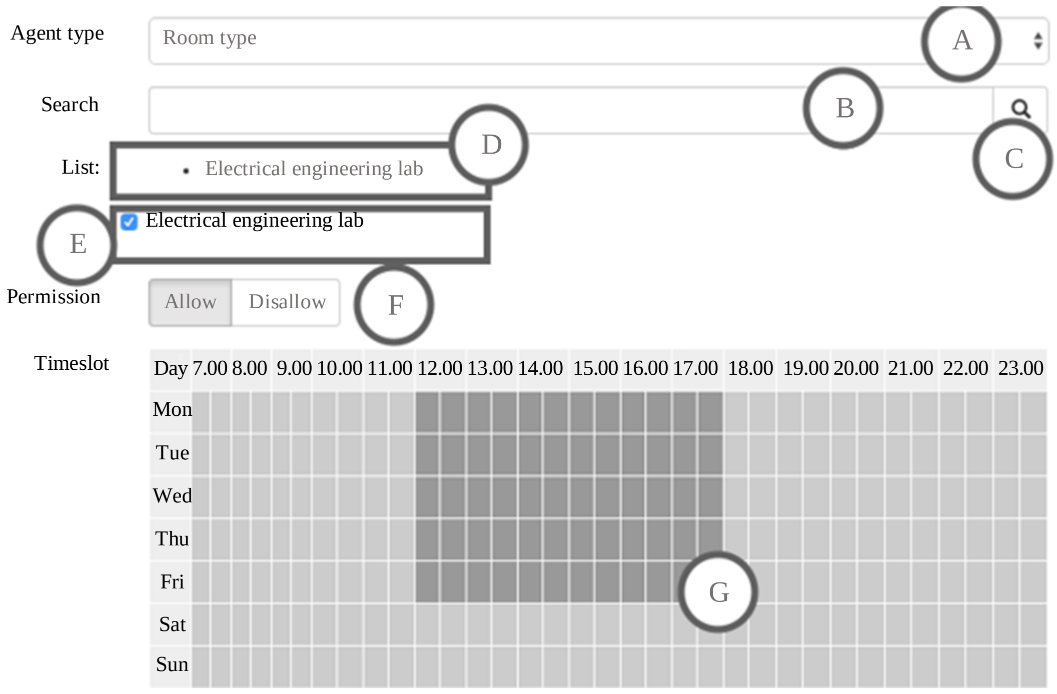
Agent type: Room type
Search:
List: Electrical engineering lab
☑ Electrical engineering lab
Permission: Allow | Disallow
Timeslot
A
B
C
D
E
F
G
Day 7.00 8.00 9.00 10.00 11.00 12.00 13.00 14.00 15.00 16.00 17.00 18.00 19.00 20.00 21.00 22.00 23.00
Mon
Tue
Wed
Thu
Fri
Sat
Sun

Fig. 4: the constraint configuration page

Conclusion

The university timetabling system is proposed in this work. The core search engine of the proposed system applies Depth-bounded Discrepancy Search (DDS) together with the heuristic and hard-soft constraints in a form of the scoring technique. To adapt to the changes in the workload, the proposed system configuration can be set by the user and the proposed system is also automatically selected based on the workload characteristics. This way, even a non-technical user can also use the proposed system easily. Moreover, the proposed system provides various output formats including the schedule of each student group, each teacher and each room.

The contributions of this work include (1) the proposed university timetabling system with a pre-defined set of recommended system configurations, (2) the workload characteristics and their effects on the proposed system and (3) the results on the modified workloads in order to provide a recommended course size to the case study university. Future works include studying the effects of the different core algorithm on the performance and the user satisfactions.

Funding Information

This research was supported by Prince of Songkla University under funding no. ENG550405S-0.

Author's Contributions

Sangsuree Vasupongayya: Contribution includes designing the research, analyzing the results, drawing the conclusions, acquiring the research funding, writing the manuscript, submitting the manuscript, contacting the editorial and reviewers.

Warakorn Sitthirit: Contribution includes researching the related literatures, coding the simulator, conducting the experiments, analyzing the results, writing the manuscript.

Suthon Sae-Wong: Contribution includes reviewing the manuscript and providing comments.

Thaniya Kaosol: Contribution includes commenting the grant proposal and manuscript.

Ethics

References

Al-Betar, M.A. and A.T. Khader, 2010. A harmony search algorithm for university course timetabling. Ann. Oper. Res., 194: 3-31. DOI: 10.1007/s10479-010-0769-z

Ameen, P., 2012. A genetic algorithm-based timetable for undergraduate engineering student. PSU, Prince of Songkla University.

Antony, E.P., C.G. Walker, M. Ehrgott and D.M. Ryan, 2017. Integer programming for minimal perturbation problems in university course timetabling. Ann. Oper. Res., 252: 283-304. DOI: 10.1007/S10479-015-2094-Z

Badoni, R.P., D.K. Gupta and P. Mishra, 2014. A new hybrid algorithm for university course timetabling problem using events based on groupings of students. Comput. Ind. Eng., 78: 12-25. DOI: 10.1016/j.cie.2014.09.020

Banbara, M., T. Soh, N. Tamura, K. Inoue and T. Schaub, 2013. For course timetabling.

Bellio, R., L. Di Gaspero and A. Schaerf, 2012. Design and statistical analysis of a hybrid local search algorithm for Course Timetabling. J. Sched., 15: 49-61. DOI: 10.1007/s10951-011-0224-2

Bellio, R., S. Ceschia, L. Di Gaspero, A. Schaerf and T. Urli, 2016a. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. Comput. Oper. Res., 65: 83-92. DOI: 10.1016/j.cor.2015.07.002

Bellio, R., S. Ceschia, L. Di Gaspero, A. Schaerf and T. Urli, 2016b. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. Comput. Oper. Res., 65: 83-92. DOI: 10.1016/J.COR.2015.07.002.

Bolaji, A.L., A.T. Khader, M.A. Al-Betar and M.A. Awadallah, 2014. University course timetabling using hybridized artificial bee colony with hill climbing optimizer. J. Comput. Sci., 5: 809-818. DOI: 10.1016/j.jocs.2014.04.002

Boland, N., B.D. Hughes, L.T.G. Merlot and P.J. Stuckey, 2008. New integer linear programming approaches for course timetabling. Comput. Oper. Res., 35: 2209-2233. DOI: 10.1016/j.cor.2006.10.016

Bonutti, A., F. De Cesco, L. Di Gaspero and A. Schaerf, 2012. Benchmarking curriculum-based course timetabling: Formulations, data formats, instances, validation, visualization and results. Ann. Oper. Res., 194: 59-70. DOI: 10.1007/s10479-010-0707-0

Burke, E.K., J. Mareček, A.J. Parkes and H. Rudová, 2010. A supernodal formulation of vertex colouring with applications in course timetabling. Ann. Oper. Res., 179: 105-130. DOI: 10.1007/s10479-010-0716-z

Chaiyasuwan, S. and W. Tantithamphusit, 2007a. The satisfaction and expectation of teachers and staff working in prince of songkla university to the learning. Teach. Exam.

Chaiyasuwan, S. and W. Tantithamphusit, 2007b. The satisfaction and expectation of student in prince of songkla university to the learning. Teach. Exam. Schedule Manage.

Chaiyasuwan, S. and W. Tantithamphusit, 2007c. Operating manual for learning, teaching and examination time-table management.

Chinnasri, W. and N. Sureerattanan, 2010. Comparison of performance between different selection strategies on genetic algorithm with course timetabling problem. Adv. Manage. Sci., 2: 105-108. DOI: 10.1109/ICAMS.2010.5552828

Fink Bagger, N.C., G. Desaulniers and J. Desrosiers, 2019. Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem. J. Sched., 22: 155-172. DOI: 10.1007/S10951-018-0582-0

Goh, S.L., G. Kendall and N.R. Sabar, 2017. Improved local search approaches to solve the post enrolment course timetabling problem. Eur. J. Oper. Res., 261: 17-29. DOI: 10.1016/J.EJOR.2017.01.040.

Gunawan, A., K.M. Ng and K.L. Poh, 2012. A hybridized Lagrangian relaxation and simulated annealing method for the course timetabling problem. Comput. Oper. Res., 39: 3074-3088. DOI: 10.1016/j.cor.2012.03.011

Jain, A., D. Jain and D. Chande, 2010. Formulation of genetic algorithm to generate good quality course timetable. Int. J. Innov., 1: 248-251.

Kohshori, M.S. and M.S. Abadeh, 2012. Hybrid genetic algorithms for university course timetabling. Int. J. Comput. Sci., 9: 446-455.

Lach, G. and M.E. Lübbecke, 2012. Curriculum based course timetabling: New solutions to Udine benchmark instances. Ann. Oper. Res., 194: 255-272. DOI: 10.1007/s10479-014-1643-1

Lewis, R., 2008. A survey of metaheuristic-based techniques for University Timetabling problems. OR Spectr., 30: 167-190. DOI: 10.1007/s00291-007-0097-0

Lü, Z. and J.K. Hao, 2010. Adaptive Tabu search for course timetabling. Eur. J. Oper. Res., 200: 235-244. DOI: 10.1016/j.ejor.2008.12.007

McCollum, B., 2007. A Perspective on Bridging the Gap Between Theory and Practice in University Timetabling. In: Practice and Theory of Automated Timetabling, Burke, E.K. and H. Rudová (Eds.), Springer, Brno, Czech Republic, pp: 3-23.

Müller, T. and H. Rudová, 2014. Real-life curriculum-based timetabling with elective courses and course sections. Ann. Oper. Res., 239: 153-170. DOI: 10.1007/s10479-014-1643-1

Müller, T. and H. Rudová, 2016. Real-life curriculum-based timetabling with elective courses and course sections. Ann. Oper. Res., 239: 153-170. DOI: 10.1007/s10479-014-1643-1

Murray, K., T. Müller and H. Rudová, 2006. Modeling and solution of a complex university course timetabling problem. Pract. Theory Autom. Timetabling, 3867: 189-209. DOI: 10.1007/978-3-540-77345-0_13

Nothegger, C., A. Mayer, A. Chwatal and G.R. Raidl, 2012. Solving the post enrolment course timetabling problem by ant colony optimization. Ann. Oper. Res., 194: 325-339. DOI: 10.1007/s10479-012-1078-5

Post, G., L. Di Gaspero, J.H. Kingston, B. McCollum and A. Schaerf, 2016. The 3rd International Timetabling Competition. Ann. Oper. Res., 239: 69-75. DOI: 10.1007/s10479-013-1340-5

Sabar, N.R., M. Ayob, G. Kendall and R. Qu, 2012. A honey-bee mating optimization algorithm for educational timetabling problems. Eur. J. Oper. Res., 216: 533-543 DOI: 0.1016/j.ejor.2011.08.006

Sánchez-Partida, D., 2013. Modeling and solving a timetabling problem considering time windows and consecutive periods. Lect. Notes Manage. Sci., 5: 25-32.

Schaerf, A. and L. Di Gaspero, 2007. Measurability and reproducibility in timetabling research: Discussion and proposals. Ractice Theory Autom. Timetabl., 3867: 40-49. DOI: 10.1007/978-3-540-77345-0_3

Sitthirit, W. and S. Vasupongayya, 2013. Applying a mixed objective model in a university timetabling solution searching technique. Proceedings of the International Computer Science and Engineering Conference, Sept. 4-6, IEEE Xplore Press, Nakorn Pathom, Thailand, pp: 266-271. DOI: 10.1109/ICSEC.2013.6694768

Song, T., S. Liu, X. Tang, X. Peng and M. Chen, 2018. An iterated local search algorithm for the University Course Timetabling Problem. Applied Soft Comput., 68: 597-608. DOI: 10.1016/J.ASOC.2018.04.034

Soria-Alcaraz, J.A., E. Özcan, J. Swan, G. Kendall and M. Carpio, 2016. Iterated local search using an add and delete hyper-heuristic for university course timetabling. Applic. Soft Comput., 40: 581-593. DOI: 10.1016/j.asoc.2015.11.043

Turabieh, H., S. Abdullah, B. McCollum and P. McMullan, 2010. Fish Swarm Intelligent Algorithm for the Course Timetabling Problem. In: Rough Set and Knowledge Technology, Yu, J., S. Greco, P. Lingras, G. Wang and A. Skowron (Eds.), Springer, Berlin, Heidelberg.

Vasupongayya, S. and S.H. Chiang, 2006. Multi-objective models for scheduling jobs on parallel computer systems. Proceedings of the International Conference on Cluster Computing, Sept. 25-28, IEEE Xplore Press, Barcelona, Spain, pp: 1-6. DOI: 10.1109/CLUSTR.2006.311873

Walsh, T., 1997. Depth-bounded discrepancy search. Proceedings of the 15th International Joint Conference on Artificial Intelligence, May 19-23, Cork, Ireland, pp: 1388-1393. DOI: 10.1007/978-3-319-07046-9

Wangthammang, T., S. Suwanmanee, T. Angchuan and S. Vasupongayya, 2018. Registration assistant using local search and tabu list technique. Thaksin Univ. J., 21: 265-283.