4th IEEE Conference on Automation Science and Engineering
Key Bridge Marriott, Washington DC, USA
August 23-26, 2008

A Variant of Examination Timetabling Problem

Weiwei Chen and Leyuan Shi

Abstract— **The examination timetabling problem is a typical combinatorial optimization problem. In literature, heuristic methods are usually chosen to solve these problems, and mathematical programming approaches have not been well developed. This paper raises a variant of exam timetabling, with multiple exam paper versions, which comes from real-world applications. The mathematical formulation is presented, and further refined under assumptions to make it solvable for large-scale problems. Then, a preprocessing procedure, called "group-and-divide", is introduced to improve the solution quality by reducing the size of the input data. In addition, the symmetry of the model is studied, and a set of constraints are added to break the strong symmetry existing in the model. Real data are tested using CPLEX solver, and the results show that our approach is effective and applicable for real-world problems.**

Index Terms— **Examination Timetabling, Mathematical Programming, Preprocessing, Symmetry Breaking**

I. INTRODUCTION

The timetabling problems arise as various real-world problems, such as educational timetabling, sports timetabling, and transport timetabling. The timetabling problem is a traditional combinatorial optimization problem. Its manual solution is usually very time costing, and the solution quality can hardly be satisfied when the problem size grows bigger. Basically, the timetabling problem consists of a sequence of scheduling activities by satisfying a set of constraints for resources. A general definition of timetabling is given by Burke, de Werra, and Kingston [1]:

> A timetabling problem is a problem with four parameters: T, a finite set of times; R, a finite set of resources; M, a finite set of meetings; and C, a finite set of constraints. The problem is to assign times and resources to the meetings so as to satisfy the constraints as far as possible.

Among different types of timetabling problems, the educational timetabling is one of the most widely studied. It usually consists of scheduling lectures or exams to teachers and/or students in a prefixed period of time, satisfying various types of constraints [2]. The educational timetabling may involve scheduling between teachers and classes [2], [3], lectures and students [1], [2], or exams and students [2], [4], [5].

As one of the classic educational timetabling, the purpose of examination timetabling (also called exam timetabling) is generally to schedule a set of exams within a given amount of time, avoiding overlap of exams having common students [2], [4]. In literature, there are both hard constraints and soft constraints for exam timetabling [5]. Hard constraints are always satisfied for the scheduling to be applicable, while soft constraints intend to make the assignment more reasonable and acceptable. Typical hard constraints include constraints for common resources such as students and rooms. No student can take two or more exams at the same time, and room capacity should always be satisfied. Soft constraints have variants. For example, each student does not take exams at consecutive time slots so that he/she will have enough time to review each exam. Not only constraints, but also the objectives of exam timetabling problems have many variants, such as to spread exams out as evenly as possible for each student [6], to minimize the total number of time periods required to accomplish all exams (especially for E-commerce classes), and etc.

Because of the variety of problem structures, different methods have been introduced to address exam timetabling. Literatures emphasize on heuristic methods. Carter and Laporte in 1996 [6] summarized the published approaches for exam timetabling as four categories, including cluster methods, sequential methods, generalized search strategies, and constraint based approaches. Cluster methods split the set of exams into groups which satisfy hard constraints and then the groups are assigned to time periods to fulfill the soft constraints [7], [8]. Sequential methods usually assign exams to a specific period one at a time based on some sequencing strategy, so that no exam in the period conflicts with each other. In sequencing methods, exam timetabling problems are usually represented as graphs [4], and graph coloring heuristics can be employed to solve the problems [6], [9]. Generalized search methods (meta-heuristics) may be the most popular choice. Over the last thirty years, many meta-heuristic approaches have been implemented. Popular search strategies are tabu search [10], [11], simulated annealing [13], and genetic algorithms [14], [15]. Some other researchers introduced constraint logic programming and constraint satisfaction techniques in solving timetabling problems [16], [17]. For comprehensive review of automated exam timetabling approaches, we refer to [5], [6], [9].

However, as mentioned in [6], global optimization approaches have not been fully developed to solve practical timetabling problems. Tripathy attempted to tackle the real problems using mathematical programming. In [18], a course timetabling problem is modeled as a binary integer linear program (ILP), and a Lagrangian relaxation approach is adopted to solve the binary ILP. This approach was then further developed and presented in [19]. However, Carter [4],

W. Chen is with the Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA wchen26@wisc.edu

L. Shi is with the Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA leyuan@engr.wisc.edu

[6] claimed that the algorithm was not a good way to solve practical timetabling problems, and further computational and theoretical research was needed. During the last two decades, the rapid development of computer technology, as well as both the theoretical and practical improvements of mathematical programming, inspires us to further seek the possibility of using mathematical approaches in timetabling.

The rest of the paper is organized as follows. Section II introduces the exam timetabling problem we are considering, and mathematical formulation are provided. In Section III, several useful procedures are presented, in order to solve practical problems using mathematical programming. In Section IV, some real data are tested using CPLEX solver, and the results are shown. Finally, Section V presents the conclusions and potential future works.

II. MODEL DESCRIPTION

As described in last section, exam timetabling have many variants, with different constraints and objectives. That's why none of the meta-heuristic methods can be applied as a framework for all timetabling problems. For example, for our specific type of problem studied throughout this paper, there is no existing heuristic methods as far as our knowledge.

This section is organized as follows. We first describe our specific exam timetabling problem in Section II-A. Then, the mathematical formulation of this problem is provided in Section II-B. In Section II-C and II-D, we make the mathematical model easier to solve by refining the formulation and relaxing some binary variables, respectively.

A. Problem Description

In traditional models, each exam is held only once during the whole exam time periods (we call it sections), and the objective is to find the schedule which minimizes the total number of time periods (sections) required. But in some cases, institutions allow an exam to be held for multiple times (at different sections). That is, an exam can be held multiple times so that students can choose to attend one of them to avoid conflict. If an exam is held at two (or more) sections, two (or more) different exam papers are used for this exam, to avoid plagiarism. We can further assume that each exam can have no more than n versions of exam papers, in order to make the schedule more reasonable. Our objective is to minimize the total number of exam papers needed, given allowed section length, which is specified by users. The constraints and objective we consider in our problem are summarized in Table I.

TABLE I
OBJECTIVE AND CONSTRAINTS IN THE MODEL

	Objective
0.	find the solution with the smallest number of total versions of exam paper used within given maximum section length
	Constraints
1.	each student should take all the exams he/she supposes to take
2.	no student can take two or more exams at the same section
3.	each exam can mostly be held n times
4.	classroom capacity constraint
5.	there should be enough classroom to hold all the exams

B. Mathematical Formulation

Sets:

- $I = \{1, \ldots, |I|\}$: the set of student id.
- $J = \{1, \ldots, |J|\}$: the set of exam id.
- $S = \{1, \ldots, |S|\}$: the set of available sections.
- $R = \{1, \ldots, |R|\}$: the set of available classrooms.

Parameters:

- $t_{ij}, i \in I, j \in J$: binary values, 1 if student i take exam j, and 0 otherwise.
- $m_r, r \in R$: the maximum number of available seats for each classroom.
- n: the maximum number of exam paper versions for each exam.

Variables:

- $x_{ijsr}, i \in I, j \in J, s \in S, r \in R$: binary decision variables, 1 if student i takes exam j at section s and classroom r, and 0 otherwise.
- $y_{jsr}, j \in J, s \in S, r \in R$: binary decision variables, 1 if exam j is held at section s and classroom r, and 0 otherwise.
- $z_{js}, j \in J, s \in S$: binary decision variables, 1 if exam j is held at section s, and 0 otherwise.

Objective:

$$\min \sum_{j \in J, s \in S} z_{js} \tag{1}$$

Constraints:

$$\sum_{s \in S, r \in R} x_{ijsr} = t_{ij} \quad \forall i \in I, j \in J \tag{2}$$

$$\sum_{j \in J, r \in R} t_{ij} \cdot x_{ijsr} \leq 1 \quad \forall i \in I, s \in S \tag{3}$$

$$\sum_{i \in I} x_{ijsr} \leq m_r \cdot y_{jsr} \quad \forall j \in J, s \in S, r \in R \tag{4}$$

$$\sum_{j \in J} y_{jsr} \leq 1 \quad \forall s \in S, r \in R \tag{5}$$

$$\sum_{r \in R} y_{jsr} \leq |R| \cdot z_{js} \quad \forall j \in J, s \in S \tag{6}$$

$$\sum_{s \in S} z_{js} \leq n \quad \forall j \in J \tag{7}$$

The objective (1) is to minimize the total number of exam papers used. Constraints (2) mean that every student should take all the exams he/she supposes to take. Constraints (3) guarantee each student cannot take more than one exam at any section. If exam j is held at section s and classroom r, the total number of students should satisfy the classroom capacity constraints (4); otherwise, no student can take exam j at section s and classroom r. Constraints (5) say that each classroom cannot hold more than one exam at the same section. If multiple classrooms are used to hold a same exam in a single section, then only one exam paper is needed, as said by constraints (6). Constraints (7) mean that there can be no more than n versions of exam paper for each exam.

By solving formulation (1)-(7), a detailed schedule for each exam and each student will be given, if the problem is solvable. However, in practical problems, it turns out that the problem sizes are too large to solve. Typically, a real problem consists of about 15,000-20,000 student, 70-100 exams, 8-10 sections, and 100-200 classrooms. We coded the model in GAMS, and set up CPLEX 10 as the solver for mixed integer programs (MIPs). In our implementations, the program failed to finish model construction, because the size of the problem is so large that the computer memory limit was reached. We can imagine that branch-and-bound and branch-and-cut algorithms can hardly handle the MIPs with such huge solution spaces. In order to make the mathematical model applicable, a refined model is further introduced to reduce the problem size.

C. Refined Formulation

The huge problem size of the above model causes mathematical programming non-applicable. The number of binary variables for the above formulation is $|I| \cdot |J| \cdot |S| \cdot |R| + |J| \cdot |S| \cdot |R| + |J| \cdot |S| \approx |I| \cdot |J| \cdot |S| \cdot |R|$. The first thought to reduce the problem size is to re-formulate the problem by defining binary variables with smaller size. From practical point of view, we notice that in institutions there are usually enough classrooms and room capacity to hold the exams. So, we assume that we don't need to worry about the room capacity, that is, constraints 3 and 4 in Table I are always satisfied. Under this assumption, we have the following refined formulation with much smaller size.

Sets:

- $I = \{1, \ldots, |I|\}$: the set of student id.
- $J = \{1, \ldots, |J|\}$: the set of exam id.
- $S = \{1, \ldots, |S|\}$: the set of available sections.

Parameters:

- $t_{ij}, i \in I, j \in J$: binary values, 1 if student i take exam j, and 0 otherwise.
- n: the maximum number of exam paper versions for each exam.

Variables:

- $x_{ijs}, i \in I, j \in J, s \in S$: binary decision variables, 1 if student i takes exam j at section s, and 0 otherwise.
- $y_{js}, j \in J, s \in S$: binary decision variables, 1 if exam j is held at section s, and 0 otherwise.

Objective:

$$\min \sum_{j \in J, s \in S} y_{js} \tag{8}$$

Constraints:

$$\sum_{s \in S} x_{ijs} = t_{ij} \quad \forall i \in I, j \in J \tag{9}$$

$$\sum_{j \in J} t_{ij} \cdot x_{ijs} \leq 1 \quad \forall i \in I, s \in S \tag{10}$$

$$\sum_{i \in I} x_{ijs} \leq |I| \cdot y_{js} \quad \forall j \in J, s \in S \tag{11}$$

$$\sum_{s \in S} y_{js} \leq n \quad \forall j \in J \tag{12}$$

Constraints (11) mean that an exam should be held at a section as long as one or more student is taking the exam at this section. Other constraints and the objective have the similar meaning to those in formulation (1)-(7).

The refined formulation (8)-(12) is proved to be good for practical use, and the room capacity assumption we made is reasonable. However, the size of the branch-and-bound tree is still large when we solve the MIP, since the number of binary variables for branching is very large, and much computational effort is wasted at unnecessary branches. This is the impetus for us to further reduce the size of binary variables.

D. Relaxing Binary Variables

We know that a MIP with feasible set $\{x \in \mathbb{R}^n : Ax \leq b, x \in \mathbb{Z}_+^n\}$ has some good properties when A is totally unimodular. Totally unimodular matrix is a matrix for which every square non-singular submatrix is unimodular (with determinant +1 or -1). We have the following theorem [20]:

Theorem 1: If A is a totally unimodular matrix and b is integral, then every extreme point of $\{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ is integral.

That is, an integer program whose constraint matrix is totally unimodular and whose right hand side is integral can be solved by linear programming (LP) since all its basic feasible solutions are integer. This conclusion is very promising, because a LP is much easier to solve than a MIP.

For the refined formulation, it can be shown that the coefficient matrix of x_{ijs} is totally unimodular and the right hand side is integral, given y_{js} are binary. So, binary variables x_{ijs} can be relaxed as continuous variables with bounds $0 \leq x_{ijs} \leq 1$. Compared to y_{js}, the size of x_{ijs} is much larger ($|I|$ times larger, where $|I|$ is usually a large number). So, by relaxing binary variables x_{ijs} as continuous variables, the only binary decision variables are y_{js}, and the size of search tree for solving the MIP should be greatly reduced. As a result, more computational time can be focused on exploring useful branches, and the solution quality can be greatly improved given limited computation time.

III. Improving Solution Quality

Solving refined formulation with some binary variables relaxed, the CPLEX MIP solver can be a powerful tool. However, the huge solution space is the biggest concern for practical problems, because people usually want to achieve good results within limited time. A preprocessing procedure and symmetry breaking constraints are introduced to better improve the solution quality within given time limit.

A. Group-and-Divide Preprocessing

In refined formulation, it is obvious that set I has the largest size (usually 15,000-20,000). Examining the practical data, there exist many students sharing exactly the same exam schedule, which usually happens in real world. If we "bundle" this "group" of students together, the optimal solution will not change because we have assumed enough room capacity. The benefit of "bundling" (also called "grouping")

is that the size of the variables x_{ijs} is greatly reduced. Actually, without affecting the optimal result, we exclude the possibility that each student in this group has different schedules. This procedure implicitly cuts off many branches in a search tree, and thus saves some computational time.

Definition 1: A *group* of students is the set of students who take exactly the same exams. Each group is unique, that is, all students who share the same exams must be in the same group.

Example 1: Student 1 and 2 take both exam A and B, student 3 takes exam B, and student 4 takes exam C. Then, we say group 1 consists of student 1 and 2, group 2 consists of only student 3, and group 3 consists of only student 4.□

The "dividing" procedure is from another point of view. It targets on separating the original problem into several subproblems, while each of them is an independent timetabling problem. If we assume that there are always enough rooms to hold students, the optimal value of the original problem is simply the sum of the optimal values of all subproblems. We can further conclude that the minimum number of sections required for the original problem is the maximum one of all subproblems. We introduce the basic idea of dividing using the following simple example.

Example 2: Student 1 takes exam A and B, student 2 takes exam A and C, student 3 takes exam B and C, student 4 takes exam D and E, and student 5 takes exam E. Then, it is apparent that we can divide students into two divisions $\{1,2,3\}$ and $\{4,5\}$, and divide exams into two divisions $\{A,B,C\}$ and $\{D,E\}$. We then define two subproblems: subproblem 1 with $I_1 = \{1,2,3\}, J_1 = \{A,B,C\}$ and subproblem 2 with $I_2 = \{4,5\}, J_1 = \{D,E\}$. Obviously, solving subproblems 1 and 2 separately is much easier than solving the original problem. □

The **dividing procedure** can be described as follows:
Step 0. Initialize the student pool I to be all available students, and subset of students $\tilde{I}$ and subset of exams $\tilde{J}$ to be empty.
Step 1. Pick a student in the student pool I. If this student takes at least one of the exams in $\tilde{J}$, add all the exams this student takes to $\tilde{J}$, add this student to $\tilde{I}$, and delete this student from I. Continue to the next student, until all students in the student pool is checked once.
- If there is no new element added to set $\tilde{I}$ and $\tilde{J}$ during this loop, go to step 2.
- Otherwise, repeat step 1 with current student pool.
Step 2. Output current set $\tilde{I}$ and $\tilde{J}$ as the data set for a new subproblem. If current student pool is empty, terminate; otherwise, empty set $\tilde{I}$ and $\tilde{J}$, and go to step 1.

To summarize, the group-and-divide procedure can be described as a preprocessing procedure manipulating the input data matrix t_{ij}. Without formal description, we introduce this feature by a simple example.

Example 3: $I = \{1,2,3,4,5\}, J = \{A,B,C,D,E,F\}$, and the t_{ij} matrix (denoted by T) is as follows:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

First, the grouping procedure combines row 1 and row 5, and the resulting matrix (denoted by G) is as follows:

$$\begin{array}{c} \\ 1' \\ 2' \\ 3' \\ 4' \end{array} \begin{array}{c} \begin{array}{cccccc} A & B & C & D & E & F \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

where 1'={1,5}, 2'={2}, 3'={3}, and 4'={4}.

Then, the dividing procedure is actually to transform matrix G into a block diagonal matrix by multiplying by certain permutation matrix P_1, P_2: $G' = P_1 G P_2$.

$$\begin{array}{c} \\ 1' \\ 4' \\ 2' \\ 3' \end{array} \begin{array}{c} \begin{array}{cccccc} A & C & E & B & D & F \end{array} \\ \left[\begin{array}{ccc|ccc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{array}\right] \end{array}$$

Thus, the resulting subproblems are: $I_1 = \{1',4'\} = \{1,4,5\}, J_1 = \{A,C,E\}$, and $I_2 = \{2',3'\} = \{2,3\}, J_2 = \{B,D,F\}$. □

For the preprocessing procedure, how good the impact will be depends on the structure of the input data. If we are lucky enough, we expect to reduce the model size greatly (at least factor 10 in our applications). From computational point of view, the group-and-divide procedure is worth trying, because it won't cause too much computational time to run this procedure. Actually, the computational cost is polynomial (the worst case is $O(|I|^2|J|)$ for grouping and also $O(|I|^2|J|)$ for dividing).

B. Symmetry Breaking

Another noticeable feature of the model is its symmetry. It could be painful to spend a lot of computational effort on unnecessary branches of search tree. Carefully comparing the original formulation and the refined one, it is not difficult to notice that the latter one has less symmetry than the former one. The original formulation actually induces symmetry by having an additional dimension (classroom), since it makes no difference for an exam to be held in different classrooms. But in this case, the MIP solver needs to explore equivalent search regions unnecessarily. This is one reason why refined formulation can achieve better performance. The grouping procedure can also been viewed as a symmetry reduction process. Each identity in a group has the same property, and many equivalent search regions are excluded by bundling them together. From a similar point of view, there still

exists certain symmetry in the refined formulation (8)-(12). Considering the following example.

Example 4: Consider a problem with two sections and three exams. One optimal solution is: exam 1 and 2 are held in section 1, and exam 3 is accomplished in section 2. That is, $y_{11} = y_{21} = y_{32} = 1$ and other binary variables equal to 0. Apparently, there is an equivalent optimal solution $y_{12} = y_{22} = y_{31} = 1$ and others 0. □

The goal of symmetry breaking is to prune the search tree in order to avoid the exploration of unnecessary search space. In literature, symmetry breaking can be achieved according to specific problem structures. In [21], Sherali and Smith discussed three applications which have symmetric solutions: a telecommunications network design problem, a noise pollution problem, and a machine procurement and operation problem. By incorporating hierarchical constraints, the effect of symmetry of each problem is mitigated. Other examples of symmetry breaking can be found in [22].

Reviewing Example 4, there exist at least two equivalent optimal solutions. One of the simple properties of the first solution is $y_{11} + y_{21} + y_{31} + y_{41} > y_{12} + y_{22} + y_{32} + y_{42}$. The physical meaning of this property can be explained as arranging the busiest section (in terms of the number of exams) as early as possible. So, for our exam timetabling problem, one simple symmetry breaking constraint is

$$\sum_{j \in J} y_{j,s+1} \leq \sum_{j \in J} y_{js} \quad \forall s \in S \backslash \{|S|\} \tag{13}$$

By adding these constraints, we will be able to avoid some of the unnecessary searches, and might achieve better result within given time limit. Another benefit of these constraints is they are simple and do not add much computational burden.

IV. TESTING RESULTS

We coded in GAMS, and CPLEX 10 is applied as the MIP solver. We have three different data sets as shown in Table II, where $|I|$ denotes the size of the student set, and $|J|$ denotes the size of the exam set. N is the number of student-exam pairs (non-zeros in all t_{ij}).

TABLE II
TESTING DATA SETS

Data	$\|I\|$	$\|J\|$	N
Set 1	20695	78	88168
Set 2	26050	78	104658
Set 3	14720	108	62329

For large MIPs, different CPLEX options will surely produce different performances, especially in those cases where optimal solutions are very hard to be found. We tried some combinations, and finally dedicate to four sets of options as shown in Table III. For detail meaning of the CPLEX options, we refer to CPLEX 10 solver manual.

All following tests are based on refined formulation with x_{ijs} relaxed. Time limit of all tests are set to be 36000 seconds (10 hours). The maximum allowed number of sections are set to be 8. The testing results are displayed in Table IV. We compare our results to the known heuristic results provided by the data provider.

TABLE III
DIFFERENT SETS OF CPLEX OPTIONS

Option	Codes
1	default CPLEX options
2	mipemphasis 2, varsel 3
3	cuts -1, flowcovers 0, fraccuts 0, implbd 0, mipemphasis 2, varsel 3
4	cuts -1, mipemphasis 2, varsel 3

TABLE IV
BEST SOLUTIONS FOR DIFFERENT CPLEX OPTIONS

Data	Opt. 1	Opt. 2	Opt. 3	Opt. 4	Heuristic
Set 1	124	120	539	116	163 (9 sections)
Set 2	512	440	121	117	161

From Table IV, we can see that our results are generally better than the results generated by the heuristic approach. Especially for data Set 1, heuristic failed to find a solution with 8 sections' limit. The biggest concern is that since the problem size is very large, it could be very expensive to generate cuts. This is the reason why for several CPLEX options our solution is bad. For these cases, much of the resource (time or memory) is used in generating cuts and doing heuristic searches, instead of to explore nodes in the branch-and-bound tree. This explains why option 4 (turning off all the cuts) usually performs better than other options.

Then, we further focused on CPLEX option 3 and 4, and tested the performances of group-and-divide procedure. The best solutions found are shown in Table V.

TABLE V
PERFORMANCE OF GROUP-AND-DIVIDE (G&D) PROCEDURE

Data	Opt. 3	Opt. 4	Opt. 3 w/G&D	Opt. 4 w/G&D
Set 1	539	116	119	112
Set 2	121	117	121	117
Set 3	203	207	186	196

From Table V, we can see that the effect of preprocessing is obvious. Actually, the grouping procedure is the main factor. It reduces the problem size to more than 10 times smaller than the original one. The dividing procedure has very limited impact for these three data sets. For data set 1 and 2, it turns out that we can not divide the original problems. For data set 3, we did find two subproblems. However, one of the subproblem is rather small (one student group takes one exam), and the other subproblem is almost the same size as the original problem. Again, if we are lucky enough, the dividing procedure can also has positive impacts for some data sets. We can also find out the fact that data set 3 is more difficult to solve (maybe because of more exams - bigger $|J|$), and cuts finally appear to be powerful (option 3 performs better than option 4). Of course, another advantage

to use preprocessing is that the current best solution is reached much faster since problem size is now smaller and each node is faster to solve. So, if we set a smaller time limit (e.g., 5 hours), we can imagine that solution quality should be much better if we use preprocessing.

The final test is to compare the effect of symmetry breaking cuts (all data are preprocessed using G&D).

TABLE VI
PERFORMANCE OF SYMMETRY BREAKING CONSTRAINTS (SBC)

Data	Opt. 3	Opt. 4	Opt. 3 w/SBC	Opt. 4 w/SBC
Set 1	119	112	122	115
Set 2	121	117	114	120

From Table VI, the best solution of data set 2 is further improved by adding constraints (13), while the best solution of data set 1 is not. But one thing to note: the best solution of data set 1 with symmetry breaking constraints and CPLEX option 4 (115) is achieved in about 5,000 seconds; the best solution of data set 2 with symmetry breaking constraints and CPLEX option 4 (120) is achieved in about 3,000 seconds. This is a relatively short time which could be set in practical implementations. This feature could be useful for the users who intend to get a good solution without waiting too long.

V. CONCLUSIONS AND FUTURE WORKS

In this paper, an exam timetabling problem with multiple exam papers is introduced. A mathematical formulation is built and refined to make it applicable for real-world problems. The features of the refined formulation are further studied. Some of the binary variables are relaxed without affecting the optimal solution, and symmetry breaking constraints are added to avoid unnecessary search in equivalent regions. Furthermore, a preprocessing procedure called group-and-divide is presented, in order to reduce the size of the problem. Three set of real data are tested, and the results show that the proposed approaches are effective and suitable for practical problems.

Several potential future research directions are as follows:

- In this paper, we are trying to improve the solution quality within given time limit. However, the gap seems to be not small enough because the solver can not find a good lower bound. The linear programming relaxation of the original problem is rather loose, and no cutting plane has found to improve the lower bound. The obvious lower bound which is adopted by CPLEX is the total number of exams ($|J|$). If we are able to generate tighter bounds, the gap can be reduced and more branches can be cut.
- As published in [18], [19], Lagrangian relaxation could be a good choice for large-scale mixed integer programs, in order to either search better solution or explore tighter bounds. Combining Lagrangian relaxation with our current approaches might produce better results for exam timetabling.
- In section III, we proposed a set of simple constraints to break the symmetry existing in the model. There might be other more complicated symmetry breaking constraints, which could be more powerful to produce better results or increase lower bounds.

REFERENCES

[1] E. Burke, D. de Werra, and J. Kingston, *Handbook of Graph Theory*. CRC Press London, 2004, ch. Applications to timetabling, pp. 445–474.

[2] A. Schaerf, "A survey of automated timetabling," *Artificial Intelligence Review*, vol. 13, no. 2, pp. 87–127, 1999.

[3] D. de Werra, "The combinatorics of timetabling," *European Journal of Operational Research*, vol. 96, no. 3, pp. 504–513, 1997.

[4] M. W. Carter, "A survey of practical applications of examination timetabling algorithms," *Operations Research*, vol. 34, no. 2, pp. 193–202, 1986.

[5] R. Qu, E. Burke, and B. McCollum, "A survey of search methodologies and automated approaches for examination timetabling," 2006, unpublished. [Online]. Available: http://citeseer.ist.psu.edu/qu06survey.html

[6] M. W. Carter and G. Laporte, "Recent developments in practical examination timetabling," in *Practice and Theory of Automated Timetabling*, ser. Springer Lecture Notes in Computer Science, E. Burke and P. Ross, Eds., vol. 1153, 1996, pp. 3–21.

[7] J. G. Fisher and D. R. Shier, "A heuristic procedure for large-scale examination scheduling problems," *Congressus Numerantium*, vol. 39, pp. 399–409, 1983.

[8] N. Balakrishnan, A. Lucena, and R. T. Wong, "Scheduling examinations to reduce second-order conflicts," *Computers and Operations Research*, vol. 19, no. 5, pp. 353 – 361, 1992.

[9] E. K. Burke and S. Petrovic, "Recent research directions in automated timetabling," *European Journal of Operational Research*, vol. 140, no. 2, pp. 266–280, 2002.

[10] A. Hertz, "Tabu search for large scale timetabling problems," *European Journal of Operations Research*, vol. 54, no. 1, pp. 39–47, 1991.

[11] L. D. Gaspero and A. Schaerf, "Tabu search techniques for examination timetabling," in *The Practice and Theory of Automated Timetabling*, ser. Springer Lecture Notes in Computer Science, E. Burke and W. Erben, Eds., 2001.

[12] D. Johnson, "Timetabling university examinations," *Journal of the Operational Research Society*, vol. 41, no. 1, pp. 39–47, 1990.

[13] J. Thompson and K. A. Dowsland, "General cooling schedules for a simulated annealing based timetabling system," in *The Practice and Theory of Automated Timetabling*, ser. Springer Lecture Notes in Computer Science, E. Burke and P. Ross, Eds., 1996.

[14] D. Corne, H. Fang, and C. Mellish, "Solving the modular exam scheduling problem with genetic algorithms," in *Proceedings of the 6th international conference on Industrial and engineering applications of artificial intelligence and expert systems*, ser. Lecture Notes In Computer Science, 1993, pp. 370–373.

[15] E. Burke, D. Elliman, and R. Weare, "A genetic algorithm based university timetabling system," in *2nd East-West International Conference on Computer Technologies in Education*, 1994.

[16] P. Boizumault, Y. Delon, and L. Peridy, "Constraint logic programming for examination timetabling," *The Journal of Logic Programming*, vol. 26, no. 2, pp. 217–233, 1996.

[17] P. David, "A constraint-based approach for examination timetabling using local repair techniques," in *Practice and Theory of Automated Timetabling*, ser. Springer Lecture Notes in Computer Science, E. Burke and M. Carter, Eds., 1998.

[18] A. Tripathy, "A lagrangean relaxation approach to course timetabling," *Journal of the Operational Research Society*, vol. 31, pp. 599–603, 1980.

[19] ——, "School timetabling - a case in large binary integer linear programming," *Management Science*, vol. 30, no. 12, pp. 1473–1489, 1984.

[20] L. A. Wolsey, *Integer Programming*. Wiley-Interscience, 1998.

[21] H. D. Sherali and J. C. Smith, "Improving discrete model representations via symmetry considerations," *Management Science*, vol. 47, no. 10, p. 13961407, 2001.

[22] T. Fahle, S. Schamberger, and M. Sellmann, "Symmetry breaking," in *Principles and Practice of Constraint Programming - CP2001*, ser. Springer Lecture Notes in Computer Science, G. Goos and J., Eds., 2001.