Journal of Mathematical Extension
Vol. 16, No. 12, (2022) (5)1-31
URL: https://doi.org/10.30495/JME.2022.2362
ISSN: 1735-8299
Original Research Paper

A New Adjustment of the Branch and Price Algorithm for University Course Timetabling

M. Khorramizadeh
Shiraz University of Technology

Abstract. In 2005 Avella and Vasil'Ev [2] presented an efficient cutting plane algorithm for solving an integer binary programming formulation of the university course timetabling problem (UCTP). Here, we present a new and efficient adjustment of the branch and price algorithm for solving the same formulation of UCTP. In every iteration of the branch and price algorithm, the column generation algorithm is used for solving the linear programming relaxation. For the first time, in this paper the set packing constraints of the UCTP formulation are chosen as the specially structured constraints of the column generation algorithm. Then, a new efficient two phase heuristic method is presented for solving the set packing problem. The resulting adjusted column generation is used within a branch and price algorithm and a comparison is performed with the cutting plane algorithm presented by Avella and Vasil'Ev. The numerical results show that the computing time of the presented branch and price algorithm is always less than that of branch and cut algorithm. Moreover, the number of subproblems and master problems of the presented branch and price algorithm depends on the structure of the problem. Finally, for all test instances the number of variables is very large that justifies the use of the branch and price algorithm for solving the problem.

Keywords and Phrases: University Course Timetabling, Branch and price, Column generation, Heuristic, Set packing.

Received: April 2022; Accepted: August 2022

1 Introduction

The field of timetabling contains important ares such as employee and sports timetabling, flight scheduling, timetabling in universities and other institutions of education. This amount of usage interested many researchers to study this problem. Timetabling has been proved to be an NP-hard problem by Evans et. al. [13] (see also Bardadym [6]), this make all computational calculations more difficult. Wren [30] defined the timetabling as the allocation of some resources to some objects subject to constraints, such that a set of objectives are satisfied.

Every university faces different variants of timetabling such as course timetabling and exam timetabling at least twice a year. This paper, is concerned with the course timetabling in universities. To define the University Course Timetabling problem (UCTP), suppose that a set of teachers, a set of courses, a set of classes, a set of rooms, a set of time periods and a time horizon are given. The UCTP is to assign courses to rooms and to time periods, satisfying additional constraints. An assignment is called feasible, if it satisfies the set of hard constraints. On the other hand soft constraints are used as the means for measuring the solution quality. Hard and soft constraints originate from the organization of an educational system.

The UCTP is usually divided into curriculum-based and enrollment-based categories. In the curriculum-based category, rooms and time periods are assigned to courses according to the curriculum specified by the university [18]. The enrollment-based course timetabling is based on the enrollment data of each individual student [21].

In this paper, an efficient branch and price algorithm is presented for solving the university course timetabling problem. In every iteration of the presented branch and price algorithm, the subproblem is solved by a new efficient heuristic algorithm for solving the set packing problem. Some numerical results are also given to evaluate and compare the presented algorithm. Numerical results justifies both the use of branch and price algorithm for solving the problem and the efficiency of the presented algorithm. Moreover, the numerical results show that the number of variable for all instances is very large. This justifies the necessity of conducting research.

1.1 Literature review

Scientific papers on UCTP can be divided into two major categories. The first category is concerned with heuristic and metaheuristic methods (see [3, 14, 23, 26, 28, 29, 22, 19]).

The second category contains exact algorithms. Since this paper is concerned with an exact algorithm, here we only review research papers of the second category. By using exact algorithms to solve UCTP, we can issue the certificates of optimality and study the quality of solutions found.

In 2004, Martin [20], an integer programming model is used to allocate teachers to courses, rooms, and time periods. Their presented model handles constraints such as back-to-back classes, maximum number of teaching days, time periods of any configuration, multiple teachers for courses, departmental balance, and pre-assignments. Their presented model suggests four–index binary variable x_{ijkl}, accepting value one if the instructor i is assigned to course j, in classroom k and in time slot l. They used the CPLEX software to quite easily solve the UCTP at Ohio University's College of Business.

Daskalaki et al. [11] developed an integer programming formulation for the university course timetabling, using the six–index binary variables $x_{i,j,k,l,m,n}$, taking value one, when course m, taught by teacher l to the group of students k, is scheduled for the jth period of day i in classroom n. Constraints of their integer programming model deals with uniqueness, completeness, consecutiveness, repetitiveness and pre-assignment constraints.

The objective was the minimization the cost of allocating courses to time periods and the cost of assignment of those courses that require sessions of more than one consecutive hours, on a given day of the week. Their presented model was solvable by software tools with integer programming solvers. Their presented approach solved the UCTP of the department of engineering that has many teachers and courses.

In 2005, Daskalaki and Birbas [12] proposed a two-stage relaxation procedure that solves efficiently the integer programming formulation of a university timetabling problem. Their presented model was also based on the six–index binary variables. In their presented approach, the relaxation was performed in the first stage and concerns the constraints that

guarantee consecutiveness in multi-period sessions of certain courses. They compared to a solution approach solving the problem in a single stage. Numerical results showed that computation time is reduced significantly without any loss in quality.

In 2005, Avella and Vasil'Ev [2] described a binary integer programming problem formulation of UCTP and developed a branch and cut algorithm for solving this formulation. The formulation was based on three-index binary variables x_{crt} taking value one, when course c is scheduled in room r at time period t. They reported a successful case-study where a branch and cut algorithm yields the optimal solution of a real-world timetabling problem for university courses. They also studied the polyhedral structure of the problem related to a polytop of the set packing problem and introduced several families of cutting planes that are crucial for finding an optimal solution for the problem.

Qualizza and Serafini [24] proposed an original approach which was essentially based on an integer linear programming formulation, but, instead of the usual assignment, binary variables for each weekly course timetable were used. Due to the exponential number of different course timetables the formulation requires a column generation scheme. Their integer programming formulation was based on two–index binary variables x_{jc} taking value one if pattern j in $P(c)$ is used for course c, where, the pattern $P(c)$ is the set of timetable patterns for the course c.

They expressed the constraints of the problem by using the integer linear programming matrix. Their objective function was the maximization of the preference. They embedded the constraints and preferences associated to a single course timetable in the column generation procedure. They also discussed the interaction between the column generation procedure and the branch-and-bound method. Finally, they used the CPLEX routines to solve their presented model.

In 2007, Al-Yakoob and Sherali [1], used four–index non-negative integer variables $x_{d,c_d,t,r}$, besides some special variables for tutorials and lab sessions, to formulate the class scheduling and timetabling problem faced at Kuwait University (KU), where, $x_{d,c_d,t,r}$ is the number of sections of class c_d that are offered by department d during time-slot t in a room of type r. Their principal focus was to design efficient class offering patterns while taking into consideration newly imposed

gender policies. They formulated a mathematical programming model that assigns offered classes to time-slots and addresses gender issues by defining appropriate surrogate constraints along with objective penalty terms. They also solved their model by using the CPLEX routines.

Schimmelpfeng and Helber [27] described an integer programming approach, which was based on a set of decision variables consisting of five–index, three–index and two–index binary or integer variables. Their model was solved by open source mixed-integer solvers and CPLEX at the School of Economics and Management at Hannover University, Germany.

In 2011, Hao and Benlic [15] presented a new partition-based approach that was based on the divide and conquer principle. Their proposed approach uses iterative tabu search to partition the initial problem into sub-problems which were solved with an ILP solver. They generated lower bounds for the curriculum-based course timetabling problem, which was presented at the International Timetabling Competition ITC–2007.

In 2012, Lach and Lubbecke [18] presented an integer programming approach to the university course timetabling problem, in which weekly lectures have to be scheduled and assigned to rooms. However, instead of directly solving a natural formulation based on three-indexed variables for the course/time/room assignment, they decomposed the problem in two stages. In the first stage, they only match time periods and lectures. In the second stage these pairs were feasibly assigned to rooms. The CPLEX was used to solve their presented model.

Burke et al. [8] suggested a branch and cut algorithm for the Udine benchmark datasets that reduced the number of variables necessary to formulate the soft constraints. They presented a branch-and-cut procedure, where constraints from enumeration of event/free-period patterns, necessary to reach optimality, were added only when they were violated. They also described problem-specific cuts from bounds implied by the soft constraints, cuts from patterns given by days of instruction and free days, and all related separation routines. They also discussed applicability of standard cuts from graph coloring and weighted matching. They used ILOG Concert and CPLEX 10 to implement their algorithm.

Gambini et al. [25] presented an integer programming formulation

for a variant of the Class-Teacher Timetabling problem, which considers the satisfaction of teacher preferences and also the proper distribution of lessons throughout the week. Their formulation was based on four–index binary decision variables x_{tcdp} getting value one if teacher t was teaching to class c at day d and the time period p. These variables were also linked to some other auxiliary variables. Their formulation contained a very large number of variables and was enhanced by cuts. Therefore, a cut and column generation algorithm to solve its linear relaxation was also provided.

Cacchiani et al. [9] splitted the objective function into two parts and formulate integer linear programming models for both. The solution of one model was obtained by using a column generation procedure. The global bound was obtained by summing up the corresponding optimal values. Their presented model was solved through the general purpose solver CPLEX.

Two MIP formulations were developed for the curriculum based course timetabling problem (CTT) by Bagger et al. [5]. They divided the CTT into two separate models and connected them by flow formulation techniques. They also showed that the resulting formulations contain underlying network structures. The first presented MIP formulation was based on the minimum cost flow problem, while the second one was based on the multi-commodity flow problem. Their numerical results showed that the first formulation performs better on average than other one for the CTT.

Bagger et al. [4], presented an integer-programming relaxation for obtaining lower bounds for the curriculum-based course timetabling problem in which weekly assignments for courses to rooms and periods are considered. Their presented model was a pattern formulation where a pattern is an assignment of a course into a set of periods on one day. Different preprocessing techniques were implemented to reduce the number of variables, and valid inequalities are derived and added to the model. The proposed model was tested on 21 real-world data instances and On 17 of these instances, the best known solutions had been proven optimal, and out of the remaining four, our model improved the lower bounds for three of them.

Colajanni and Daniele [10], presented an integer programming for-

mulation for the curriculum-based course timetabling problem. They also applied their presented model to the real case study of the first year of the Mathematics Degree Course of the University of Catania, Italy.

The strength of all aforementioned research papers is that they are very efficient for solving the specific problem that they are designed to solve. The main weakness of the above research papers is that they can not be applied or adapted easily to solve general course timetabling problems, efficiently. Here, we concentrate on the integer programming formulation presented by Avella and Vasil'Ev [2]. The first reason is that this model is closer to the course timetabling problem in universities of Iran. On the other hand, in these problems, the number of variables is usually very large that suggests using the branch and price algorithm for solving this problem. The second reason is that the subproblem of the branch and price algorithm is a set packing problem that can be efficiently solved by some fast heuristic methods.

To explain the main ideas of this manuscript, at first we need to briefly explain the column generation algorithm and the set packing problem.

1.2 Column generation algorithm

In this section we briefly describe a generic column generation algorithm. Consider the following linear programming problem:

$$\begin{array}{lcll} Minimize & c^T x & & \\ s.t. & Ax & = & b \\ & x & \in & X \end{array} \qquad (P)$$

where, $A \in R^{m\times n}$, $c \in R^n$ and $b \in R^m$ and X is a polyhedral set having a special structure. The set of constraints $Ax = b$ are called the general constraints and the set of constraints represented by $x \in X$ are called specially structured constraints. According to the representation theorem in linear programming, any point $x \in X$ can be written as:

$$x = \sum_{j=1}^{t} \lambda_j x_j + \sum_{j=1}^{l} \mu_j d_j, \qquad \sum_{j=1}^{t} \lambda_j = 1, \qquad \lambda_j \geq 0, \quad \mu_j \geq 0. \qquad (1)$$

where $x_1, \ldots, x_t$ are extreme points of X and $d_1, \ldots d_l$ are extreme directions of X. If we substitute for x in (1), the previous optimization

problem is transformed into the next master problem in the variables $\lambda_1, \ldots, \lambda_t$ and $\mu_1, \ldots, \mu_l$.

$$\begin{array}{lllll} Minimize & \sum_{j=1}^{t}(c_j^T x_j)\lambda_j & +\sum_{j=1}^{l}(c_j^T d_j)\mu_j & & \\ s.t. & \sum_{j=1}^{t}(Ax_j)\lambda_j & +\sum_{j=1}^{l}(A^T d_j)\mu_j & = b & (MP) \\ & \sum_{j=1}^{t}\lambda_j = 1, & \lambda_j \geq 0, & \mu_j \geq 0. & \end{array}$$

In every iteration of the column generation, given a basic feasible solution having w as the vector of dual variables, we solve the following so-called subproblem which is easy because of the special structure of X:

$$\begin{array}{lll} Max & (w^T A - c^T)x & \\ s.t. & x \in X. & (SP) \end{array}$$

If the optimal value of this problem is zero, then the current basic feasible solution is optimal. Otherwise, a candidate variable enters the basis. In a column generation algorithm, the major computational effort is due to solving the pricing problem. In other words if we can solve the pricing problem efficiently, then the resulting column generation algorithm is also efficient.

1.3 Set packing problem

In the set packing problem (SPP), there are n objects which can be packed into a number of subgroups among m predefined feasible subsets labeled as $S_1, \ldots, S_m$. Each subset S_j has a payoff value of w_j. The SPP aims to divide these n objects into non-overlapping subgroups such that their total payoff is maximized. The integer binary programming model for the set packing problem is as follows:

$$\begin{array}{lll} max & \sum_{j=1}^{n} w_j x_j & \\ & \sum_{j=1}^{n} a_{ij} x_j \leqslant 1, & i = 1, \ldots, m, \\ & x_j \in \{0, 1\}, & \forall j = 1, \ldots, n. \end{array}$$

where the a_{ij} are 0/1 coefficients and the w_j are weights.

1.4 Main ideas of this paper

In this paper, we consider the same formulation as the one proposed by Avella and Vasil'Ev [2], (to be described in Section 2) and present a new

and efficient adjustment of the branch and price algorithm for UCTP. In every iteration of the branch and price algorithm, a linear programming relaxation is solved by using the column generation algorithm.

On the other hand, in the binary integer programming formulation of Avella and Vasil'Ev [2], many constraints of the UCTP are in the form of the constraints of the set packing problem. Therefore, to make the column generation algorithm as efficient as possible, we first consider set packing constraints of the UCTP as specially structured constraints and the rest of constraints of UCTP as the general constraints. Then, we present an efficient two phase heuristic algorithm for solving the set packing problem which is used for solving the pricing problem in every iteration of the column generation algorithm. The resulting column generation algorithm will be used for solving the pricing problem, in every iteration of the branch and price algorithm. Finally, we compare the resulting branch and price algorithm with the branch and cut algorithm of Avella and Vasil'Ev [2]. Finally, we present numerical results that justify the efficiency of our presented algorithm.

Recall that the column generation based research papers (that we are aware of), are those of Qualizza and Serafini [24], Gambini et al. [25] and Cacchiani et al. [9]. Some important differences between these three approaches and the approach of this paper, will be illustrated in the following sections and during the presentation of the contents of this paper.

In Section 2 we describe the binary integer programming formulation of UCTP presented by Avella and Vasil'Ev [2]. In Section 4 we present our new adjustment of the branch and price algorithm for solving UCTP. Section 5 is concerned with the numerical results justifying the efficiency of our presented branch and price algorithm for solving UCTP. Section 5 is devoted to conclusions.

2 Binary Integer Programming Formulation

This paper is concerned with the integer programming formulation of the problem presented by Avella and Vasil'Ev [2].

To describe this formulation, at first we need to consider some notations. Let C be the set of courses and $\bar{c}$ be the number of elements

of C ($|C| = \bar{c}$). For $c \in C$, let n_c and n^c_{min} (n^c_{max}) denote the number of hours to be scheduled per week and the minimum (maximum) daily number of teaching hours. Let G be a set of groups (a group consists of students attending exactly the same courses). Suppose that $|G| = \bar{g}$ and for $g \in G$, let $C_g \subset C$ denote courses attended by group g. It is assumed that some classes can attend the same courses. Let D be the working days of the week. let $|D| = \bar{d}$ and for any $d \in D$, let τ_d and ι_d denote the first time periods of the morning and afternoon session in day d, respectively.

Let S be the set of teachers. Assume that $|S| = \bar{s}$ and for $s \in S$, let $C_s \subset C$ denote the set of courses taught by teacher s and k_s be the maximum number of teaching days. Let R and T be the set of rooms and time periods, respectively. Moreover, suppose that $|R| = \bar{r}$, $|T| = \bar{t}$, l_{max} is the maximum daily number of teaching hours for any groups of students and p_{crt} is the penalty of scheduling course $c \in C$, in room $r \in R$ and at the time period $t \in T$. Let the binary variable x_{crt} be 1 if course $c \in C$ is scheduled in room $r \in R$ at time period $t \in T$. Suppose that the binary variable u_{cd} is 1 if the course $c \in C$ is assigned to the day $d \in D$ and the binary variable ψ_{sd} is 1 if $d \in D$ is a teaching day for teacher $d \in D$.

A feasible (acceptable) assignment of courses C to rooms R and to time periods T, should satisfy the following conditions. n_c hours of a week must be scheduled for every course $c \in C$ i.e.

$$\sum_{r \in R} \sum_{t \in T} x_{crt} = n_c, \qquad c \in C. \tag{2}$$

Each class $g \in G$ cannot attend more than one course at a time period $t \in T$,

$$\sum_{c \in C_g} \sum_{r \in R} x_{crt} \leqslant 1, \qquad g \in G, t \in T. \tag{3}$$

Each teacher s cannot teach more than one course at time $t \in T$,

$$\sum_{c \in C_s} \sum_{r \in R} x_{crt} \leqslant 1, \qquad s \in S, t \in T. \tag{4}$$

Each room $r \in R$ cannot host more than one course at time $t \in T$,

$$\sum_{c \in C} x_{crt} \leqslant 1, \qquad r \in R, t \in T. \tag{5}$$

The number of hours of the course $c \in C$ is day $d \in D$, must be in $[n^c_{min}, n^c_{max}]$ i.e.

$$\sum_{r \in R} \sum_{\tau_d \leqslant t < \tau_{d+1}} x_{crt} \geqslant n^c_{\min} u_{cd}, \qquad c \in C, d \in D. \tag{6}$$

$$\sum_{r \in R} \sum_{\tau_d \leqslant t < \tau_{d+1}} x_{crt} \leqslant n^c_{\max} u_{cd}, \qquad c \in C, d \in D. \tag{7}$$

The timetable of each course should be compact.

$$\sum_{r \in R} (x_{crt_1} - x_{crt_2} + x_{crt_3}) \leqslant 1 \quad \begin{array}{c} c \in C, d \in D, \\ \tau_d \leqslant t_1 < t_2 < t_3 < \tau_{d+1}. \end{array} \tag{8}$$

All the hours of a course $c \in C$ scheduled in a day $d \in D$ should be located in the same room $r \in R$.

$$x_{cr_1t_1} + x_{cr_2t_2} \leqslant 1, \quad \begin{array}{c} c \in C, 1 \leqslant r_1 < r2 \leqslant \bar{r}, \\ d \in D, \tau_d \leqslant t_1 < t_2 < \tau_{d+1}. \end{array} \tag{9}$$

Each group cannot attend more than l_{max} teaching hours a day.

$$\sum_{c \in C_g} \sum_{r \in R} \sum_{\tau_d \leqslant t < \tau d+1} x_{crt} \leqslant l_{\max}, \qquad g \in G, d \in D \tag{10}$$

Courses of a group can be taught only either in the morning or in the afternoon session.

$$\sum_{r \in R} x_{c_1rt_1} + \sum_{r \in R} x_{c_2rt_2} \leqslant 1 \quad \begin{array}{c} g \in G, c_1, c_2 \in C_g, c_1 \neq c_2, d \in D \\ \tau_d \leqslant t_1 < \iota_d \leqslant t_2 < \tau_{d+1} \end{array} \tag{11}$$

The timetable of each group should be compact.

$$\sum_{c \in C_g} \sum_{r \in R} (x_{crt_1} - x_{crt_2} + x_{crt_3}) \leqslant 1 \quad \begin{array}{c} g \in G, d \in D, \\ \tau_d \leqslant t_1 < t_2 < t_3 < \tau_{d+1}. \end{array} \tag{12}$$

The number of working days of the teacher $s \in S$ must be less than or equal to k_s,

$$\sum_{r \in R} x_{crt} \leqslant \Psi_{sd}, \qquad c \in C, s \in S, d \in D, \tau_d \leqslant t < \tau_{d+1}. \tag{13}$$

$$\sum_{d \in D} \Psi_{sd} \leqslant k_s, s \in S. \tag{14}$$

Each course $c \in C$ can only be assigned to a specified subset of rooms $R_c \subset R$.

$$x_{crt} = 0, \quad c \in C, r \in R \diagup R_c, t \in T. \tag{15}$$

Each room is available only in a subset of timeslots $T_r \in T$,

$$x_{crt} = 0, \quad c \in C, r \in R, t \in T \diagup T_r. \tag{16}$$

Each teacher is available only in a subset of timeslots $T_s \in T$,

$$x_{crt} = 0, \quad s \in S, c \in C_s, r \in R, t \in T \diagup T_s. \tag{17}$$

Finally, the objective is to minimize the sum of the penalties,

$$\min \sum_{c \in C} \sum_{r \in R} \sum_{t \in T} p_{crt} x_{crt}. \tag{18}$$

3 Comparison with Other Column Generation Approaches

Here, we explain some differences between our presented approach and other approaches based on column generation. In Qualizza and Serafini [24] the integer programming formulation was based on two–index binary variables x_{jc} taking value one if pattern j in $P(c)$ is used for course c, where, the pattern $P(c)$ is the set of timetable patterns for the course c. In Gambini et al. [25], the formulation is based on four–index binary decision variables x_{tcdp} getting value one if teacher t was teaching to class c at day d and the time period p. These variables were also linked to some other auxiliary variables.

In Cacchiani et al. [9], variables are of two types. A variable of the first type represents a feasible assignment of lectures to rooms and time periods for the whole time horizon. A variable of the second type represents a feasible assignment of lectures to time periods for the whole time horizon, such that it only takes into account the penalties deriving from violating the minimum number of working days and having isolated lectures. Therefore, the variables of this paper are quite different from those of [9, 25, 24].

To clarify the difference between the model of this manuscript and the model of other three approaches using the column generation more, note that, Qualizza and Serafini [24], did not handle the set of constraints (7), (8), (10) and (12). Gambini et al. [25], did not handle the set of constraints (7), (8), (10), (11) and (12). Cacchiani et al. [9], did not handle the set of constraints (7), (8), (10) and (11). Moreover, Qualizza and Serafini [24], considered as objective function the maximization of a preference. In Cacchiani et al. [9] and Gambini et al. [25], the objective function aims at minimizing the total cost of the penalties for violating the soft and hard constraints.

4 Adjustment of the Branch and Price Method

In this section we describe our presented branch-and-price algorithm for university course timetabling problem. In a branch and price algorithm, the LP-based branch-and-bound algorithm is combined with the column generation algorithm to solve each LP relaxation. In the column generation algorithm, the pricing subproblems should be fast to solve.

In every iteration of the branch and price algorithm, the linear programming relaxation of the binary integer programming formulation of the UCTP described in section 2 is solved by using the column generation algorithm. Therefore, if we present an efficient algorithm for solving the column generation algorithm then we can expect that the resulting branch and price algorithm is also efficient.

The efficiency of the column generation algorithm strongly depends on the way we partition the constraints of UCTP into general constraints and specially structured constraints, and on the efficiency of the algorithm solving the pricing problem. In our presented approach, the

specially structured constraints are chosen such that the resulting subproblem is a set packing problem. Indeed, the set of constraints (3), (4), (5), (9) and (11) are considered as the specially structured constraints and the remaining constraints are chosen as general constraints. Recently, we presented an efficient two phase algorithm (to be described in the next subsection) for solving the set packing problem and justified its efficiency. In this paper we use this algorithm for solving the set packing problem corresponding to the pricing problem in every iteration of the column generation algorithm.

4.1 Two phase algorithm for solving set packing problem

The algorithm has two phases. In the first phase, a greedy method is utilized for generating a set of feasible solutions P_0. The second phase is concerned with an iterative step. In the iterative step some members of P_0 are chosen, a combination procedure is applied to them and new solutions replace worst members of the population. The iterative step is repeated until the stopping criteria are satisfied. For set packing problem, we use a binary vector $p \in \{0,1\}^n$ as a solution representation. For example, suppose that $n = 6$. A binary vector $p = (0,1,0,0,1,1)$ is a binary vector in $\{0,1\}^6$ and this solution corresponds to the selection of second, fifth and sixth subsets in the solution.

4.1.1 Phase one: construction of a set of feasible solutions

In this subsection we describe the greedy algorithm for generating a set of ps distinct feasible solutions $P_0 = \{p_1, p_2, \ldots, p_{ps}\}$. At first we describe a quick overview of the algorithm. The algorithm starts by sorting the variables using a predetermined priority (to be described later). Then, ps feasible solutions are generated. Each feasible solution is generated from the trivial solution by iteratively setting a variable's value to 1.

Moreover, this greedy algorithm has ps iterations. In each iteration, a solution is built from the trivial feasible solution, $x_j = 0$, $1 \leq j \leq n$. Some variable values are set to 1, as long as the solution is maintained feasible. Changes concern only one variable at each iteration. Variables with a maximum coefficient in the objective value are prioritized and the one with highest priority is chosen as the variable that is set to 1.

The tie breaking rule is as follows. For the first variable, among the most interesting variables, the choice is the one corresponding to the one involving a minimum number of constraints and with the minimum index (in order).

For the rest of variables, among the most interesting variables, the choice is the one corresponding to the one with the minimum index. Here, experimentally, we let the tie breaking rule for selecting the first variable to be more expensive than other variables. Changes stop when no variable can be fixed to 1 without losing feasibility. The pseudo-code of the algorithm for generating initial population is presented in Algorithm 1. In step 1 of this algorithm, the population P_0 is initialized. In the step 2, variables are ordered according to their coefficient in the objective value and are stored in list L. While steps 1 and 2 comprise the initial steps, steps 3 to 11 are concerned with iterative steps of Algorithm 1. In the ith iteration of Algorithm 1, the set of nonzero variables $\bar{S}_i$ and the set of infeasible variables Z_i are initialized to empty set. In step 5, the first member of $\bar{S}_i$ is determined as described before. In step 6, the selected member is added to $\bar{S}_i$ and Z_i, D_k and T_i are updated accordingly. The rest of members of $\bar{S}_i$ are determined through steps 7 to 10. More specifically, in step 8, the variable with largest weight in the objective function outside $\bar{S}_i \cup Z_i$ is selected. In step 9, this variable is inserted to $\bar{S}_i$ and Z_i is updated. In step 11, the constructed solution is added to P_0.

Extensive numerical results presented in [17] show that algorithm 1 generate feasible solutions with high quality. In the ith iteration of this algorithm, $\bar{S}_i$ is the set of nonzero variables in the ith solution and if a variable in Z_i is set to one then the solution becomes infeasible.

4.1.2 Phase two: iterative step

In phase two of the algorithm an iterative step is repeated until the stopping criteria are satisfied. In the iterative step, at first two members q_1 and q_2 of the initial set of solutions are randomly chosen. Then, a combination method is used to generate a new solution q_{new} as follows.

Let $I_1 = \{x_j : x_j = 1 \text{ in } q_1\}$ and $I_2 = \{x_j : x_j = 1 \text{ in } q_2\}$. The combination method starts with the feasible solution in which all variables are zero. Then all variables in $I = I_1 \bigcap I_2$ are set to one. Here,

Algorithm 1 Phase I: Generating an initial set of feasible solutions

1: $P_0 \leftarrow \varnothing$.
2: $L \leftarrow$ List of sorted variables based on their coefficient in the objective function.
3: **for** i:=1 **to** ps **do**
4: $\bar{S}_i \leftarrow \varnothing$ and $Z_i \leftarrow \varnothing$.
5: Let x_k correspond to the first member of L such that $x_k \notin \bigcup_{t=0}^{i-1} \bar{S}_t$. Among the most interesting variables, choose the one corresponding to the one involving a minimum number of constraints and with the minimum index (in order).
6: $\bar{S}_i \leftarrow \bar{S}_i \cup \{x_k\}$. $Z_i = Z_i \cup D_k$, where $D_k = \left\{x_t : t \in \bigcup_{i \in I_k} T_i\right\}$ and $T_i = \{j : a_{ij} = 1, 1 \leq j \leq n\}$.
7: **for** $j := 2$ **to** n **do**
8: Let x_k correspond to the largest weight in objective function such that $x_k \notin \bar{S}_i \cup Z_i$.
9: $\bar{S}_i \leftarrow \bar{S}_i \cup \{x_k\}$. $Z_i = Z_i \cup D_k$.
10: **end for**
11: Add the new constructed solution to P_0.
12: **end for**

note that since q_1 and q_2 are feasible the resulting solution is also feasible. To maintain the feasibility, for all x_k in I, all variables in $D_k = \left\{x_t : t \in \bigcup_{i \in I_k} T_i\right\}$ are added to the set Z_c (all variables in Z_c are not set to 1 in the combination method).

To complete the combination method, we have to decide whether variables outside of I have value one or not. Therefore, we use the following iterative procedure. In every iteration four different candidate variables are examined and the one corresponding to the greatest improvement in the objective function, will be chosen and is set to one in q_{new}. The set of these four candidate variables consists of a variable in $(I_2 \setminus I_1) \bigcap Z_c^c$, a variable in $(I_1 \setminus I_2) \bigcap Z_c^c$, and two variables in $(I_1 \bigcup I_2) \bigcap Z_c^c$, where Z_c^c denotes the complement of Z_c.

These variables are selected as follows. The first one is a variable with greatest weight in $(I_2 \setminus I_1) \bigcap Z_c^c$. Here, if there are more than one variable with greatest weight in $(I_2 \setminus I_1) \bigcap Z_c^c$, then the one with smallest

number of nonzero in the corresponding column of A, is chosen. If there are still more than one variable with these properties, then the variable with smallest index is chosen. The second candidate variable is a variable in $(I_1 \setminus I_2) \bigcap Z_c^c$ and is chosen similarly. The third candidate variable is chosen in a similar way from $(I_1 \bigcup I_2) \bigcap Z_c^c$. Finally, the fourth variable is randomly chosen from $(I_1 \bigcup I_2) \bigcap Z_c^c$ such that it is different from the third one.

After the selection of one of these candidate variables, the value of the selected variable x_r will be changed from zero to one in q_{new} and is added to $\bar{S_c}$. Then, all variables in $D_r = \left\{x_t : t \in \bigcup_{i \in I_r} T_i\right\}$ are added to Z_c. The aforementioned iterative procedure stops when no variable outside $\bar{S_c}$ can be set to 1 without destroying the feasibility. The pseudo-code of the combination method is presented in algorithm 2. Step 1 indicates that the input of this algorithm is the set of initial population generated by Algorithm 1. Steps 2 to 15 comprise the iterative steps of Algorithm 2. In every iteration of Algorithm 2 at first two random members of P_0 are selected in step 3. In step 4, the set of nonzero variables $\bar{S_c}$ and the set of infeasible variables Z_c are initialized to empty set. The set of all nonzero variables in the selected solutions are added to $\bar{S_c}$. Z_c is updated accordingly in step 6. Steps 7 to 13 are concerned with the inner loop of Algorithm 2, where all components are processed and the new solution is constructed. Finally, the new solution produced by using the combination method replaces the worth member of the set of solutions in step 14.

Here, we explain another difference between our presented approach and the column generation based approaches of Qualizza and Serafini [24], Cacchiani et al. [9] and Gambini et al. [25]. In none of the other three approaches, the subproblem is a set packing problem. Moreover, the other three approaches use the CPLEX which is a General-purpose software. Our presented approach uses a Special–purpose software designed to solve the set packing problem.

In what follows, we describe the presented specific branch and price algorithm. The original problem (OP) described in section 2 can be

Algorithm 2 Phase II: Iterative step

Let P_0 be the set of feasible solutions constructed in phase I.
while Stopping criteria are not satisfied **do**
  Select two random solutions q_1 and q_2.
  $\bar{S}_c \leftarrow \varnothing$, $Z_c \leftarrow \varnothing$.
  Add all variables in $I_1 \cap I_2$ to $\bar{S}_c$.
  $\forall x_k \in I_1 \cap I_2$, add all variables in $D_k = \left\{x_t : t \in \bigcup_{i \in I_k} T_i\right\}$ to Z_c.
  while $\left|\bar{S}_c \cup Z_c\right| < n$ **do**
    Select a variable $x_{\alpha_1} \in (I_2 \setminus I_1) \bigcap Z_c^c$ with greatest weight.
    Select a variable $x_{\alpha_2} \in (I_1 \setminus I_2) \bigcap Z_c^c$ with greatest weight.
    Select a variable $x_{\alpha_3} \in (I_1 \bigcup I_2) \bigcap Z_c^c$ with greatest weight.
    Randomly, select a variable $x_{\alpha_4} \in (I_1 \bigcup I_2) \bigcap Z_c^c$.
    Let x_k be the best variable from of the set $\{x_{\alpha_1}, x_{\alpha_2}, x_{\alpha_3}, x_{\alpha_4}\}$. Add x_k to the set of $\bar{S}_c$. Add all variables in $D_k = \left\{x_t : t \in \bigcup_{i \in I_k} T_i\right\}$ to Z_c.
  end while
  Replace the worst solution in P_0 by the constructed solution.
end while

written as follows:

$$min \quad c^T x \tag{19}$$
$$A_1 x = b_1, \quad \text{constraints (2)}, \tag{20}$$
$$A_2 x \geq b_2, \quad \text{constraints (6)}, \tag{21}$$
$$A_3 x \leq b_3, \quad \text{constraints (7), (8), (10), (12), (13), (14)}, \tag{22}$$
$$x \in H, \quad \text{constraints (3), (4), (5), (9), (11)}. \tag{23}$$

Since H is a bounded polyhedron we have:

$$H = \{x : \ x = \sum_{t=1}^{T} \lambda_t x^t, \ \sum_{t=1}^{T} \lambda_t = 1, \ \lambda_t \in \{0,1\}, \ \text{for } t = 1, \dots T\}.$$

Therefore, OP can be written as the following Integer Master Problem

(IMP):

$$
\begin{aligned}
&min \quad \sum_{t=1}^{T}(c^T x^t)\lambda_t,\\
&\sum_{t=1}^{T}(A_1 x^t)\lambda_t = b_1, \sum_{t=1}^{T}(A_2 x^t)\lambda_t \geq b_2, \sum_{t=1}^{T}(A_3 x^t)\lambda_t \leq b_3,\\
&\sum_{t=1}^{T}\lambda_t = 1, \quad \lambda_t \in \{0,1\}, \quad t = 1,\dots T.
\end{aligned}
$$

The presented branch and price algorithm at first applies the column generation algorithm (to be described later) on the linear programming relaxation of IMP (LPRIMP). Let $\tilde{\lambda} = (\tilde{\lambda}_1, \dots, \tilde{\lambda}_T)$ be the optimal solution of LPRIMP. If $\tilde{\lambda}$ is not integer, then the IMP is not yet solved. To solve the IMP we use the ideas of the branch and bound algorithm as follows. Since points $\eta^t \in H$ are 0–1 vectors, we have $\tilde{\eta} = \sum_{t=1}^{T}\lambda_t \eta^t \in \{0,1\}^T$ if and only if $\tilde{\lambda}$ is integer. Therefore, if $\tilde{\lambda}$ is not integer, then there is some j that $\tilde{\eta}_j$ is fractional and on which we can branch. This way the set X of all feasible solutions is split into $X_0 = X \cap \{\eta : \eta_j = 0\}$ and $X_1 = X \cap \{\eta : \eta_j = 1\}$ and the ideas of the branch and bound algorithm can be applied. Two new active nodes are added to the search tree of branch and bound and a new integer master problem is associated to each new active node. Next an active node of the search tree is selected and the corresponding linear programming relaxation is solved by using the column generation algorithm. The resulting algorithm that combines the ideas of the column generation an branch and bound is called the branch and price algorithm. The pseudocode of the specific branch and price algorithm is presented in Algorithm 4.

Next we describe the column generation algorithm. The column generation is used to solve the linear programming relaxation of the integer master problem (LPRIMP) associated to each node the search tree in the branch and price algorithm. In the column generation algorithm, at first a subset of columns of LPRIMP is selected that provide us with a feasible Restricted Master Problem (RMP). To select these columns the phase I or the big–M method can be used (see [7] for details).

Now, assume that we are in the ith iteration of the column generation and the columns corresponding to each $\lambda_t \in T_i$ have been added to the RMP. Therefore, the RMP has the following form:

$$\begin{aligned} min \quad & \sum_{t\in T_i}(c^T x^t)\lambda_t, \\ & \sum_{t\in T_i}(A_1 x^t)\lambda_t = b_1, \sum_{t\in T_i}(A_2 x^t)\lambda_t \geq b_2, \sum_{t\in T_i}(A_3 x^t)\lambda_t \leq b_3, \\ & \sum_{t\in T_i}\lambda_t = 1, \quad \lambda_t \geq 0, \quad t \in T_i. \end{aligned}$$

Then, we use the CPLEX software to solve the RMP and obtain the optimal primal solution $\bar{\lambda}^*$ and optimal dual solution $(\pi_1^*, \pi_2^*, \pi_3^*, \mu^*)$. In the next step, we consider the pricing problem (subproblem)

$$\zeta_i = max\{(c^T - (\pi_1^*)^T A_1 - (\pi_2^*)^T A_2 - (\pi_3^*)^T A_3)x - \mu^* : \ x \in H\}.$$

To solve the above subproblem we first apply the presented two phase algorithm (Algorithm 2) to obtain a near optimal basic feasible solution $\tilde{\eta}$. Then we apply the simplex algorithm on $\tilde{\eta}$ to obtain an optimal basic feasible solution of the subproblem. Let ζ_i be the optimal solution. If $\zeta_i = 0$ then $\bar{\lambda}^*$ is optimal for LPRIMP. If $\zeta_i > 0$ then the column corresponding to the optimal solution is introduced that leads to a new RMP. The pseudo-code of the column generation is presented in Algorithm 3.

Algorithm 3 Column Generation

1: (**Initialization**) Select a subset of columns providing a feasible Restricted Linear Programming Master Problem (RMP).
2: (**Solving RLPMP**) Solve the RLPMP to obtain optimal primal solution $\bar{\lambda}^*$ and optimal dual solution $(\pi_1^*, \pi_2^*, \pi_3^*, \mu^*)$.
3: (**Solve the subproblem**) Apply the two phase algorithm (Algorithms 1 and 2) for solving the set packing problem to obtain a nearly optimal basic feasible solution $\tilde{\eta}$. Use the simplex method starring from $\tilde{\eta}$, to compute the optimal basic feasible solution of the subproblem. Let ζ be the optimal solution of the subproblem.
4: (**Stopping criterion**) If $\zeta = 0$ then stop $\bar{\lambda}^*$ is optimal.
5: (**Generating a new column**) If $\zeta > 0$ add the column corresponding to the optimal solution to the RLPMP and go to step 2.

Algorithm 4 Specific Branch and price algorithm

1: (**Initialization of the root**) Associate the original problem S^0 to the root node of the search tree and add the root node to the list of active nodes of search tree.
2: (**Process the root**) Apply the column generation method (Algorithm 3) to solve the linear programming relaxation of the original problem S^0 with formulation P^0. If the optimal solution (with optimal value $\overline{z}_0$) is feasible (and therefore optimal) for S^0, then Stop. Let $\lambda^* = \text{NULL}$ and $GUB = +\infty$ (λ^* and GUB denote the best feasible solution and objective value obtained so far). i=1.
3: **while** The list of active nodes of search tree is not empty **do**
4: (**Node selection**) Select an active node with associated problem S^i and formulation P^i from the search tree.
5: (**Bounding**) Apply the column generation algorithm (Algorithm 3) to solve the linear programming relaxation of S^i with formulation P^i.
6: (**Prune by infeasibility**) If the problem is infeasible, then prune by infeasibility.
7: (**Prune by bound**) Let λ^i and llb be the optimal solution and optimal value of linear programming relaxation of S^i. If $llb > GUB$ then prune by bound.
8: (**Prune by optimality**) If the optimal solution of linear programming relaxation of S^i is integer, then update the best feasible solution and objective value obtained so far (if $llb < GUB$ let $GUB = llb$ and $\lambda^* = \lambda^i$).
9: (**Branching**) Let $\tilde{\eta} = \sum_{t=1}^{T} \lambda_t \eta^t \in \{0,1\}^T$. Select an index j such that $\tilde{\eta}_j$ is fractional. Select an index j such that η_j is fractional. Branch on η_j (Two new active nodes are added to the search tree). on which we can branch
10: **end while**

5 Numerical results

In this section we examine the numerical performance of the branch and price method. All calculations and methods run on a system with Win 7 operating system with Intel(R) Core(TM) i3 of CPU and 6 Gb of RAM. The presented branch and price algorithm of this paper was compared with the branch and cut algorithm of Avella and Vasil'Ev [2]. Because, both algorithms are exact algorithms and are based on the same model. The branch and cut algorithm of Avella and Vasil'Ev [2] was re-implemented and both algorithms were run to optimality. The experiment aims to evaluate the presented branch and price algorithm on 21 public competition instances of the Track 3 of ITC–2007 [16]. The characteristics of these problem instances are given in table 1. In this table, "name" denotes the name of the problem, $\bar{c}$ denotes the number of courses, $\bar{r}$ denotes the number of rooms, $\bar{t}$ denotes the number of time periods and $\bar{g}$ denotes the number of groups.

Clearly, there are some differences between the UCTP problem of this paper and those of Track3 of ITC–2007. However, we needed some test problems to discuss the efficiency of our presented method and perform a comparison with the branch and cut method of Avella and Vasil'Ev [21]. Therefore, we decided to use the ITC–2007 instances.

Here, we explain how we used an instance of Track 3 of ITC–2007 to test algorithms of this paper. In an instance of Track 3 of ITC–2007 each course c has a certain number of lectures, a minimum number of working days (mw_c) and a number of students attending this course. On the other hand, to test algorithms of this paper, each course of an input instance must have a weekly number of teaching hours (n_c) and a minimum (maximum) number of daily teaching hours ($n^c_{min}(n^c_{max})$), a subset of rooms $R_c \subset R$ to which the course c can be assigned.

Therefore, for each course of an input instance of Track 3 of ITC–2007, we let n_c be the number of lectures, n^c_{min} be one and n^c_{max} be the greatest integer less than or equal to n_c/mw_c i.e. $\lfloor n_c/(mw_c) \rfloor$. Moreover, we let R_c be those rooms for which the number of students of the course c is less than or equal to the capacity of the room. We let l_{max} and k_s (for each teacher $s \in S$) be the number of days of a week. We assumed that all rooms are available at all time periods. Time periods at which the teacher s is available ($T \diagup T_s$) are exactly those of Track 3 of ITC–2007.

In an instance of Track 3 of ITC–2007, there are no morning and afternoon sessions. For some instances that was possible, we divided daily time periods into morning and afternoon sessions. These instances are recorded in table 1. In this table "name" denotes the name of the instance, "All" denotes the number of all time periods in a day, "morning"("afternoon") denotes the number of time periods that we considered for morning (afternoon) session. For the rest of instances we assumed that we have only one session in every day. Note that, because of the above differences, it is not possible to perform a comparison between our results and the best results of other authors on the Track3 of ITC–2007.

name	All	morning	afternoon
comp11	9	1 −5	6 −9
comp12	6	1 −3	4 −6
comp18	6	1 −3	4 −6

Table 1: Morning and Afternoon sessions of some instances

Without loss of generality, all coefficients of the variables in the objective function are set to one. However, these coefficients can be changed according to the desirability of the corresponding university for which the scheduling is performed. The numerical results are recorded in table 2. In this table nv denotes the number of variables, nm denotes the number of master problems in the branch and price method and ns denotes the number of subproblems. Moreover, $time$ denotes the computing time (in minutes) and obj denotes the objective function value and $O.M$ means that the corresponding algorithm runs out of memory.

In table 2, our presented branch and price method is compared with the branch and cut method of Avella and Vasil'Ev [2], with respect to the computing time and the objective function value. The numerical results of this table show that in all cases, the computing time of our presented branch and price method is less than that of the branch and cut method. This happens because in every iteration of the branch and cut method a general linear programming relaxation is solved, while in every iteration of our presented branch and price method the sub-

problem is solved with an efficient algorithm that is designed to solve the corresponding set packing problem. In some cases (comp07 and comp20) the branch and cut method fails to obtain the solution of the problem, while our presented branch and price algorithm obtains the optimal solution in a reasonable computing time. However, in some cases (comp12,comp15,comp16) both problems fail to obtain the optimal solution of the problem. What caused the algorithms to fail on some instances was the lack of system memory. In all cases, both algorithms obtain the same objective function value.

								Branch and Price		Branch and Cut	
	name	$\bar{c}$	$\bar{r}$	$\bar{t}$	$\bar{g}$	nv	ns	time(m)	obj.	time(m)	obj.
	comp01	30	6	30	14	5430	1320	48	160	83	160
	comp02	82	16	25	70	32882	3925	221	281	306	281
	comp03	72	16	25	68	28872	3625	167	253	196	253
	comp04	79	18	25	57	35629	3625	243	270	384	270
	comp05	54	9	36	139	17550	7020	111	152	157	152
	comp06	108	18	25	70	48708	4375	259	357	554	357
	comp07	131	20	25	77	65631	4900	647	434	o.m	–
	comp08	86	18	25	61	38786	3875	213	324	311	324
	comp09	76	18	25	75	34276	4025	219	277	268	277
90	comp10	115	18	25	67	51865	4325	598	370	729	370
	comp11	30	5	30	13	6780	1890	56	162	59	162
	comp12	88	11	36	150	34936	8460	–	–	–	–
	comp13	82	19	25	66	39032	4050	254	308	305	308
	comp14	85	17	25	60	36210	3625	203	275	280	275
	comp15	72	16	25	150	28872	3625	–	–	–	–
	comp16	108	20	25	68	54108	4500	–	–	–	–
	comp17	99	17	25	70	42174	4175	296	339	353	339
	comp18	47	9	36	52	15275	3888	97	138	126	138
	comp19	74	16	25	66	29674	3700	168	248	215	248
	comp20	121	19	25	78	57569	4800	483	390	o.m	–
	comp21	94	18	25	78	5430	1320	53	327	69	327

Table 2: Comparison with the branch and cut algorithm

Figure 1: Number of variables increases with the number of courses

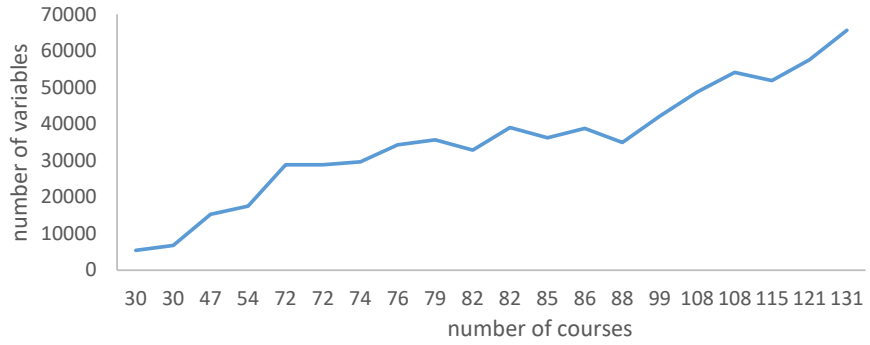

Figure 1 depicts change in the number of variables versus change of number of courses. According to this figure, the number of variables considerably increases with the number of courses. For example, when the number of courses is 131, the number of variable is 65631 i.e. the ratio between the number of courses and number of variables is approximately 0.002. This indicates that the number of variables in the instances is very large and justifies the use of branch and price algorithm for solving the problem.

Figure 2: Number of subproblems/master problems versus number of courses

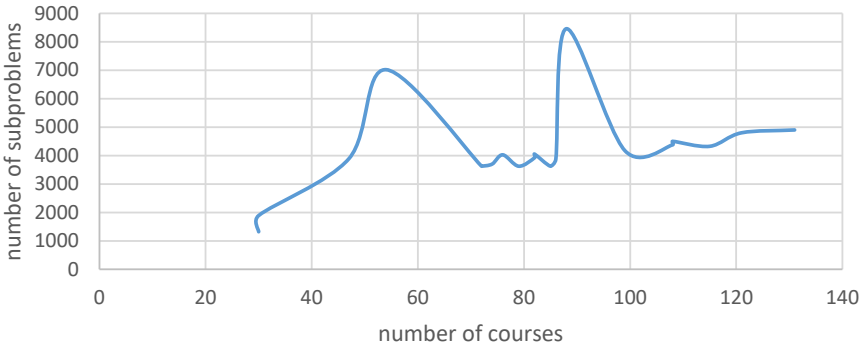

Figure 2 shows the number of generated subproblems (or equivalently master problems) for each problem instance. Form this figure, it can be observed that the number of generated subproblems does not necessarily increases with the number of courses. For example, for the problem with 88 courses the number of generated subproblems is 8460, while this number for the problem with 131 courses is 4900. Therefore, the number of generated subproblems depends on the structure of the problem.

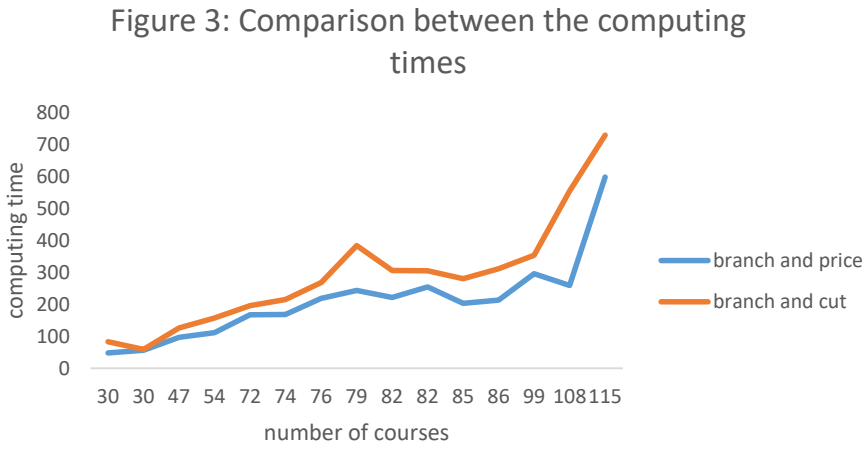

Figure 3 is concerned with the computing time of the branch and price algorithm and branch and cut algorithm. The numerical results show that for all instances the computing time of the branch and price algorithm is considerably less than that of the branch and cut algorithm. Moreover, for larger instances the difference between these computing times is larger.

6 Conclusions

In this paper, we considered a binary integer programming formulation of the university course timetabling problem and presented an efficient adjustment of the branch and price algorithm for solving the problem. Then, we performed a comparison with the branch and cut algorithm of Avella and Vasil'Ev [2]. The numerical results justify the efficiency of our presented algorithm. Moreover, for all test problems the computing

time of the presented branch and price algorithm was less than that of branch and cut algorithm. The number of variables compared to the size of input instances was very large that justifies the use of branch and price algorithm for solving the problem. The size of the search tree in the presented branch and price algorithm depends on the structure of the problem and does not increase necessarily with the size of the problem. One suggestion for future work is to apply metaheuristic algorithm for solving the problem and compare the results. Another suggestion is to apply other heuristic algorithms for solving the subproblem.

Acknowledgments
The author thanks the council of Shiraz University of technology for supporting this work.

References

[1] S. M. Al-Yakoob and H. D. Sherali, A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations, *European Journal of Operational Research*, 180(3) (2007), 1028-1044.

[2] P. Avella and I. Vasil'Ev, Computational study of a cutting plane algorithm for University Course Timetabling, *Journal of Scheduling*, 8 (2005), 497-514.

[3] H. Babaei, J. Karimpour and A. Hadidi, A Survey of approaches for university course timetabling problem, *Computers and Industrial Engineering* 86 (2014), 43-59.

[4] N. F. Bagger, G. Desaulniers and J. Desrosiers, Daily course pattern formulation and valid inequalities for the curriculum-based course timetabling problem, *Journal of Scheduling*, 22 (2019), 155-172.

[5] N. F. Bagger, S. Kristiansen, M. Sorensen and T. J. R. Stidsen, Flow formulations for curriculum-based course timetabling, *Annals of Operations Research*, 280 (2019), 121-150.

[6] V. A. Bardadym, 1996. Computer-aided school and university timetabling: The new wave. In: Burke, E., Ross, P. (Eds.), Practice

and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 1153, Springer, Berlin, pp. 22-45.

[7] M. S. Bazara, J. J. Javis and H. D. Sherali, *Linear Programming and Network Flows*, John Wiely and Sons, New Jersey (2010).

[8] E. K. Burke, J. Marecek, A. J. Parkes and H. Rudová, A branch-and-cut procedure for the Udine course timetabling problem, *Annals of Operations Research*, 194(1) (2012), 71-87.

[9] V. Cacchiani, A. Caprara, R. Roberti and P. Toth, A new lower bound for curriculum-based course timetabling, *Computers and Operations Research*, 40(10) (2013), 2466-2477.

[10] G. Colajanni and P. Daniele, A new model for curriculum-based university course timetabling, *Optimization Letters*, 15 (2021), 1601-1616.

[11] S. Daskalaki, T. Birbas and E. Housos, An integer programming formulation for a case study in university timetabling, *European Journal of Operational Research* 153 (1) (2004), 117-135.

[12] S. Daskalaki and T. Birbas, Efficient solutions for a university timetabling problem through integer programming, *European Journal of Operational Research*, 160(1) (2005), 106-120.

[13] S. Even, A. Itai and A. Shamir, On the complexity of timetable and multi-commodity flow problems, *SIAM Journal on Computing* 5 (1976), 691-703.

[14] C. W. Fong, H. B. Asmuni, B. McCollum, P. McMullan and S. Omatu, A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems, *Information Sciences* 283 (2014), 1-21.

[15] J. Hao and Benlic U., Lower bounds for the itc-2007 curriculum-based course timetabling problem, *European Journal of Operational Research*, 212(3) (2011), 464-472.

[16] *http* : *//www.cs.qub.ac.uk/itc*2007/*Login/SecretPage.php*

[17] M. Khorramizadeh and E. Motaghi, A Two–Phase heuristic method for solving the set packing problem, Submitted manuscript.

[18] G. Lach and M. Lubbecke, Curriculum based course timetabling: new solutions to Udine benchmark instances, *Annals of Operations Research*, 194 (2012), 255-272.

[19] A. Lemos, F. S. Melo, P. T. Monteiro, and I. Lynce, Room usage optimization in timetabling: a case study at universidade de Lisboa, Operations Research Perspectives, 6 (2019), 100092.

[20] C. H. Martin, Ohio university's college of business uses integer programming to schedule classes. *Interfaces*, 34(6) (2004), 460-465.

[21] C. Nothegger, A. Mayer, A. Chwatal, and G. R. Raidl, Solving the post enrollment course timetabling problem by ant colony optimization, *Annals of Operations Research*, 194(1) (2012), 325-339.

[22] N. Oladejo, A. Abolarinwa, S. Salawu, O. Bamiro, A. Lukman, and H. Bukari, Application of optimization principles in classroom allocation using linear programming, International Journal of Mechanical Engineering and Technology (IJMET), 10(1) (2019), 874–885.

[23] P. Pongcharoen, W. Promtet, P. Yenradee and C. Hicks, Stochastic Optimization Timetabling Tool for university course scheduling, *International Journal of Production Economics* 112 (2008), 903-918.

[24] A. Qualizza and P. Serafini, 2005. A column generation scheme for faculty timetabling. In: Burke E. and Ross P. (Eds.), Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 3616, Springer, Berlin, pp. 161-173.

[25] H. G. Santos, E. Uchoa, L. S. Ochi and N. Maculan, Strong bounds with cut and column generation for class-teacher timetabling, *Annals of Operations Research* 194(1) (2012), 399-412.

[26] A. Schaerf, A survey of automated timetabling, *Artificial Intelligence Review* 13(2) (1999), 87-127.

[27] K. Schimmelpfeng and S. Helber, Application of a real-world university-course timetabling model solved by integer programming, *OR Spectrum*, 29 (2007), 783-803.

[28] T. Song, Liu S., X. Tang, X. Peng, and M. Chen, An iterated local search algorithm for the university course timetabling problem, *Applied Soft Computing*, 68 (2018), 597-608.

[29] R. O. Vrielink, E. Jansen, E. W. Hans, and J. Van Hillegersberg, Practices in timetabling in higher education institutions: a systematic review, Annals of Operations Research, 275(1) (2019), 145–160.

[30] A. Wren, 1996. Scheduling, timetabling and rostering—A special relationship. In: Burke, E., Ross, P. (Eds.), Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science, vol. 1153. Springer, Berlin, pp. 46-75.

Mostafa Khorramizadeh
Department of Mathematical Sciences
Assistant Professor of Mathematics
Shiraz University of Technology
Shiraz, Iran
E-mail: m.khorrami@sutech.ac.ir