BJ O&PM Brazilian Journal of Operations & Production Management

ISSN 2237-8960 (Online)

ABEPRO ASSOCIAÇÃO BRASILEIRA DE ENGENHARIA DE PRODUÇÃO

RESEARCH PAPER

Hybrid VNS-TS heuristics for University Course Timetabling Problem

Dalessandro Soares Vianna[1], Carlos Bazilio Martins[1], Thiago Jéffery Lima[2], Marcilene de Fátima Dianin Vianna[1], Edwin Benito Mitacc Meza[1]

[1]Federal Fluminense University – UFF, Institute of Science and Technology – ICT, Department of Computer Science, Rio das Ostras, RJ, Brazil.

[2]Federal Fluminense University – UFF, Nova Friburgo, RJ, Brazil.

How to cite: Vianna, D.S., Martins, C.B., Lima, T.J. et al. (2020), "Hybrid VNS-TS heuristics for University Course Timetabling Problem", *Brazilian Journal of Operations & Production Management*, Vol. 17, No. 2, e2020683. https://doi.org/10.14488/BJOPM.2020.014

ABSTRACT

Goal: Propose hybrid heuristics combining VNS and Tabu Search (TS) for adding adaptability to University Course Timetabling Problem (UCTTP) resolution.

Design / Methodology / Approach: VNS and TS metaheuristics were used isolated and in combination in order to verify which shape acquired the best solutions. Those heuristics were verified using constraints found at two undergraduate courses of Federal Fluminense University. Effectiveness is verified by comparing them with solutions manually elaborated by undergraduate course coordinators.

Results: The computational results showed the efficiency of the hybrid VNS-TS heuristics, with emphasis on the heuristic VNS-TS1, which uses tabu search as local search method.

Limitations of the investigation: This research was applied using data from two undergraduate courses of Federal Fluminense University. New experiments on data from other universities will soon be carried out.

Practical implications: The hybrid heuristic VNS-TS1 has been in use since the first half of 2019 by some of UFF undergraduate courses coordinations. Even though each university has specific constraints on its timetable, the use of the proposed heuristics is possible because it has been developed using the framework FINESS that allows for the easy insertion and removal of constraints.

Originality / Value: We have the following innovative contributions: Development of heuristics with a high degree of adaptation to the needs of educational institutions; Two proposed VNS-TS heuristics; Proposed VND-TS improvement method.

Keywords: Hybrid Heuristics; UCTTP; VNS; Tabu Search; VND-TS.

1. INTRODUCTION

In recent decades, the scientific community has been struggling to find automated solutions to the process of resource allocation under constraints (Babaei et al., 2015). One of the typical problems of this nature is the scheduling problem, also known as timetabling problem, that can be found in companies, schools, universities or hospitals.

In the educational context, the timetabling problem is the task of scheduling professors and students meeting without violating their constraints (Schaerf, 1999). Lima (2016) argues that these tasks usually demand substantial time to be accomplished due to several factors that make it difficult to solve this problem manually, such as the number of students, number of courses, type of institution and their constraints.

Financial support: None.
Conflict of interest: The authors have no conflict of interest to declare.
Corresponding author: dalessandrovianna@id.uff.br
Received: 14 Dec 2018.
Approved: 30 Sep 2019.
Editor: Osvaldo L. G. Quelhas.

In Pereira and Costa (2016) is presented three categories of educational timetable scheduling: (i) school timetabling scheduling which regards scheduling for a period of time (Birbas et al., 2009); (ii) university course timetabling problem (UCTTP) which allow joining classes for efficiency (Avella and Vasil'ev, 2005); and (iii) exams timetabling problem which schedules exams in a short period of time (Kahar and Kendall, 2010). All categories have distinct constraints that depends on institution features (Asratian and Werra, 2002; Lima, 2016).

Lindahl et al. (2018) define the UCTTP as the problem of assigning courses to rooms and time periods. This scheduling should be done without violating the hard constraints – constraints that when violated make the solution infeasible – and with minimal violations of soft constraints, which do not make the solution infeasible but reduce its quality.

UCTTP is an NP-complete problem (Socha et al., 2003; Mazlan et al., 2019), which is usually solved by heuristics (Babaei et al., 2015; Schaerf, 1999). Lewis (2008), Qu and Burke (2009), Burke et al. (2010), Jat and Yang (2011), Al-Betar and Khader (2012), Abdullah et al. (2012), Bolaji et al. (2014), Bellio et al. (2016), Abdelhalim and El Khayat (2016), Pillay and Özcan (2019) and Mazlan et al. (2019) are examples of using heuristics for UCTTP. All of them are quite efficient but applied to a single institution.

Babaei et al. (2015) presents most common constraints found at UCTTP, which may be grouped as hard constraints and soft constraints. Author argues constraints differ among universities. Even when constraints are shared among universities, they can regard them as hard or soft. Werra (1985) also argues a generic approach to UCTTP seems not feasible due to distinct constraints among institutions.

To overcome this difficulty, the need arises for a dynamic tool that allows the adaptation of the restrictions to the interests of the institutions.

Thus, the principal objective of this work is to propose a flexible method that allows to insert and easily remove constraints in order to adapt the solutions to the needs of the institutions. For this purpose, two hybrid heuristics based on Variable Neighborhood Search (VNS) and Tabu Search (TS) for UCTTP were developed and adapted. Success of using metaheuristics VNS and Tabu Search on NP-Hard problems is widespread in the literature, as can be seen in Jat and Yang (2011), Zeng et al. (2016) and Delgoshaei et al. (2018).

As innovative contributions of this work, we mention:

(i) development of heuristics with a high degree of adaptation to the needs of educational institutions;
(ii) two proposed VNS-TS heuristics; and
(iii) proposed VND-TS improvement method.

The study is structured as following: Section 2 describes subjects that support the work – the problem addressed is detailed; Metaheuristics VNS and Tabu Search, used in the proposed solutions, are described; The framework FINESS, which supported the development of the proposed heuristics, is presented; and the multicriteria method AHP, used to define the violation penalties of each constraint, is introduced –; The methodology is described in Section 3; Section 4 presents the proposed heuristics for the addressed problem; Computational experiments are presented in Section 5. Finally, the conclusion and references are presented.

2. THEORETICAL FRAMEWORK

This section describes the problem being addressed, presents the metaheuristics VNS and Tabu Search, introduces the framework *FINESS*, which simplifies heuristics implementation, and presents the method AHP (Analytic Hierarchy Process).

Addressed problem

The university course timetabling problem (UCTTP) consists of scheduling university courses to rooms and time periods (Lindahl et al., 2018). In the scientific literature, UCTTP is solved by exact

methods (Phillips et al., 2017; Prabodanie, 2017; Lindahl et al., 2018; Gülcü and Bulkan, 2019) or heuristics. Because it is an NP-Hard problem (Socha et al., 2003) – the time complexity required to solve the problem grows exponentially with the size of the problem – and because it commonly has a large number of decision variables, the use of heuristic methods is the most appropriate strategy for your solution (Babaei et al., 2015). As examples of works applying heuristics to the problem addressed, we can mention: Abdelhalim and El Khayat (2016) and Jat and Yang (2011) use genetic algorithms – the second on is a hybrid version that uses tabu search as a search intensification method –; Simulated annealing is the method applied by Bellio et al. (2016); Qu and Burke (2009) and Pillay and Özcan (2019) use hyper-heuristics approaches; Harmony search algorithm is the method chosen by Al-Betar and Khader (2012) in their work; Bolaji et al. (2014) and Mazlan et al. (2019) use colony algorithms, which uses, respectively, artificial bee colony algorithm and ant colony optimization.

This work aims to solve the UCTTP of two undergraduate courses at Federal Fluminense University (UFF): Computer Science and Industrial Engineering. These courses are composed of classes and UFF's Rules of Procedure implies that each class should be divided in 2-hours allocations. These classes must start from 7 a.m. to 1 p.m. in the morning, from 2 p.m. to 6 p.m. in the afternoon and from 6 p.m. to 10 p.m. in the evening. Classes are organized in semester groups. Each class belongs to a semester group following an undergraduate course program. Classes are offered at a limit number of classrooms and at certain time periods: morning, afternoon and evening. Some classes must be offered at a specific time, i.e. they cannot be scheduled. At the beginning of each semester, students point out which classes they desire to attend.

Besides UFF's Rules of Procedure, professors and students were interviewed to catch other constraints. The list of constraints for the problem addressed is:

- Class allocated time - classes with more than 2 hours can be allocated in distinct days but they should be allocated at the same time in a week.
- Class allocated day - classes should not be offered at consecutive days in a week;
- Class maximum hours per day – refers to the maximum number of hours that a class can be assigned in a day. Depending on the penalty associated with a violation of this constraint, quality of a solution may be affected mildly or even border the impracticality;
- Class minimum hours per day - classes, if assigned in a day, must have a minimum number of hours;
- Class available shift - classes should be allocated according to the shifts (morning, afternoon or evening) available to respective university course;
- Class fixed timeslots - there are some predefined classes which are scheduled in given timeslots;
- Class interval - lunch break is 1-2 p.m.;
- Class maximal and minimal timeslots - start time of classes is 7 a.m. and ending time is 10 p.m;
- Classroom available – the number of available classrooms must be considered;
- Class semester group time collision - classes belonging to a semester group in a course program cannot be allocated at a same day and time;
- Class shift - classes offered at distinct days should be offered at the same shift (morning, afternoon or evening);
- Class time windows - classes with more than 2h in a day must be offered continuously;
- Professor unavailable time – classes must be allocated considering timeslots a professor declares himself available. However, a professor can declare timeslots of total availability, timeslots of total unavailability and timeslots of partial availability, that is, in which the professor is available, but if possible, allocation of classes should be avoided;
- Professor time collision - two or more classes cannot be assigned to a professor at a same day and time;
- Professor maximum days – refers to the maximum number of days in a week a professor has classes assigned. Depending on the penalty associated with a violation of this constraint, quality of the solution may be affected mildly or even border the impracticality;

- Professor time windows – aims to maximize the number of continuously allocated classes for a professor in a day;
- Professor class collision - a professor could not attend two classes at the same time;
- Professor hours per day - refers to the maximum number of hours a professor teaches in a day. Depending on the penalty associated with a violation of this constraint, quality of the solution may be affected mildly or even border the impracticality;
- Student classes collision – aims to maximize the number of classes each student desires to attend.

As suggested in Pereira and Costa (2016), constraints were classified as hard (must be satisfied) and soft (should be satisfied). Hard constraints caught were:

- Class semester group time collision;
- Class fixed timeslots;
- Class interval;
- Class maximal and minimal timeslots;
- Professor unavailable time – classes cannot be assigned to professor unavailable timeslots;
- Professor time collision;
- Professor class collision.

Soft constraints were divided into three groups because they may vary from desired constraints to crucial ones. Group 1 contains soft constraints with less importance. Group 2 has soft constraints a bit more important than Group 1. Group 3 contains soft constraints with higher importance, which should be satisfied whenever possible.

- Group 1
  - R1.1 - Student classes collision;
  - R1.2 - Professor maximum days – refers to the least strong case of this constraint in which, if the maximum number of days is violated the quality of solution is decreased, but it is still feasible.
  - R1.3 - Class maximum hours per day – when the maximum number of hours is violated the quality of the solution is decreased, but it is still feasible;
  - R1.4 - Class allocated time;

- Group 2
  - R2.1 - Class allocated day;
  - R2.2 - Class available shift;
  - R2.3 - Professor unavailable time – regards allocation of classes that should be avoided due to professor time constraints;
  - R2.4 - Professor time windows;
  - R2.5 - Class maximum hours per day – maximum number of hours should be considered at most. However, if this number is violated quality of the solution is greatly decreased, but it is still feasible;
  - R2.6 - Class shift;

- Group 3
  - R3.1 - Class minimum hours per day;
  - R3.2 - Class maximum hours per day – maximum number of hours should be considered at most. However, if this number is violated the quality of the solution is greatly decreased, but it is still feasible;
  - R3.3 - Classrooms available;
  - R3.4 - Class time windows;

- R3.5 - Professor maximum days – refers to the case that the maximum number of days should be considered at most. However, if this number is violated the quality of the solution is greatly decreased, but it is still feasible.

Some soft constraints are equivalent but with distinct weights (penalties). For instance, R3.5 and R1.2 deal with the maximum number of days allocated to a professor. R1.2 has an amount of days as a parameter that, if not violated, brings a bit more quality to the solution, whereas R3.5 has the maximum number of days that should be allocated. So, R3.5 has greater importance than R1.2.

Babaei et al. (2015) points out UCTTP has on average between 6 and 7 hard constraints and between 9 and 10 soft constraints. The UCTTP addressed in this paper has 7 hard constraints and 15 soft constraints, which demonstrates the complexity of this problem.

Variable Neighborhood Search

In metaheuristic Variable Neighborhood Search (VNS) (Mladenovic and Hansen, 1997), an initial solution is built by a constructive method, which is systematically disturbed using different neighborhood structures. This process acts as a search diversification mechanism. After this process a local search is performed which is typically a search intensification mechanism. Whenever local search cannot improve the best solution, the disturbance mechanism is altered using another neighborhood structure.

Although metaheuristics VNS is commonly used to solve NP-Hard problems (Burke et al., 2010; Cruz et al., 2012; Schermer et al., 2019; Cai and Zhu, 2019), only one paper – Borchani et al. (2017) – using this metaheuristic for UCTPP has been found in the scientific literature[1], which does not apply VNS directly; applies only the local search procedure on variable neighborhood – VND (variable neighborhood descent).

Figure 1 presents the standard algorithm of metaheuristic VNS for a minimization problem, which receives as input parameters the *r* neighborhood structures to be analyzed. Initially, Line 1, an initial solution, *s*, is constructed by a purely greedy method. The loop in Lines 2-14 ensures that VNS iterations will be performed until the stop condition is reached. Each of the VNS iterations begins, Line 3, with the random choice of a neighbor solution, *s*', of *s* in the current neighborhood structure, N_k. Then, Line 4, *s*' is refined by a local search method. If the solution *s*" found by refinement method is better than *s*, then *s* is updated and N_1 becomes the current neighborhood structure. Otherwise, the next neighborhood structure (N_k+1) is investigated. Finally, the best solution found, *s*, is returned in Line 15.

[1] Search performed on the SCOPUS database on 9/1/2019.

```
Procedure VNS (N1, N2, ..., Nr)
01. Build an initial solution s with a greedy constructive algorithm;
02. While stopping condition is not met do
03.    . k ← 1
04.     While k ≤ r do
05.         Randomly pick a neighbor s' in the Nk neighborhood;
06.         Apply a local search on s', generating the solution s'';
07.         If f(s'') < f(s) then
08.             s ← s'';
09.             k ← 1;
10.         Else
11.             k ← k+1;
12.         End-If
13.     End-While
14. End-While
15. Return s;
END-VNS
```

Figure 1. Standard algorithm of metaheuristic VNS (Mladenovic and Hansen, 1997).

Tabu Search

The metaheuristic Tabu Search – TS – (Glover, 1989; 1990) is an adaptive auxiliary procedure that guides a local search algorithm in continuous exploration within a search space. Starting from an initial solution, the search moves, at each iteration, to the best solution in the neighborhood, not accepting moves that lead to already visited solutions, which are stored in a tabu list[2]. The list remains in memory by saving solutions already visited (tabu) over a number of iterations (tabu time). As a result, the global optimum is expected to be found or a solution very close to it.

TS is one of the most commonly used local search-based metaheuristics for solving NP-Hard problems (Zeng et al., 2016; Delgoshaei et al., 2018; Schermer et al., 2019). This fact also occurs reviewing the scientific literature on the UCTPP, where TS is often used in hybrid versions (Jat and Yang, 2011; Lohpetch and Jaengchuea, 2016; Feng et al., 2017).

Figure 2 presents the standard tabu search algorithm for a minimization problem. In Line 1, a solution is constructed by a purely greedy method. The loop in Lines 4-11 ensures that the search will remain until the stopping criterion is met. In Line 5, the best neighbor solution *s'* generated from a non-tabu movement or satisfying an aspiration criterion[3] is selected. In Line 06, the tabu list *T* is updated. The conditional test on Lines 8-10 verifies that the best solution *s** should be updated. Finally, line 12 returns the best solution found.

[2] Limited size list that stores solutions already visited by the algorithm.
[3] Criterion that allows the occasional acceptance of a tabu move.

```
Procedure Tabu Search
01. Build an initial solution s with a greedy constructive algorithm;
02. s* ← s;
03. T ← Ø;
04. While stopping condition is not met do
05.     Let s' be the best solution next to s generated by a non-taboo movement or
        satisfying an aspiration criterion;
06.     Update the tabu list T;
07.     s ← s';
08.     If f(s) < f(s*) then
09.         s* ← s;
10.     End-If
11. End-While
12. Return s*;
End-Tabu Search
```

Figure 2. Standard algorithm of metaheuristic Tabu Search (Glover, 1989).

Framework FINESS

FINESS (Framework for the Implementation of metaheuristics based on Neighborhood Structure Search) (Vianna et al., 2014) is a framework for programming heuristics based on local search metaheuristics (GRASP, ILS, Simulated Annealing, VNS and Tabu Search), which is under development by the Fluminense Federal University.

Research to develop an environment that facilitates the development of heuristics is the focus of some studies in the literature. Some computational tools for this purpose already exist to reduce heuristic development complexity and make them more generic. Among the available tools are: iOPT (Voudouris et al., 2001), EO (Keijzer et al., 2001), JEO (Arenas et al., 2002), Hot-Frame (Fink and Voß, 2002), HeuristicLab (Wagner and Affenzeller, 2005), EasyLocal ++ (Di Gaspero and Schaerf, 2003), ParadisEO (Cahon et al., 2004), JCLEC (Ventura et al., 2008), OPT4J (Lukasiewycz et al., 2011), MDF (Lau et al., 2007) and Jenes (Troiano and Pasquale, 2010). These tools have in common the goal of reusing code by providing implemented common parts in metaheuristic developments. Thus, only the specific parts of a new heuristic need to be implemented by the user.

The framework FINESS, like the others computational tools, focuses on reuse, but also on other features:

- Hibridization - it can combine distinct metaheuristics for providing a solution to a problem;
- Parallelism - it can take profit of computation infrastructure and metaheuristics features for implementing faster solutions;
- Reuse - it allows reuse among distinct metaheuristics implementation;
- Extensibility - it allows extensions in two levels:
    - macro level regards adding metaheuristics to FINESS without side effects;
    - micro level regards adding new constraints and new neighborhood structures;
- Performance - besides being easy to use, FINESS aims performance when compared with manual implementation.

Among those characteristics, extensibility concerns FINESS framework's main goal. Class diagram of this framework is shown in Figure 3. The objective of this figure, in this paper, is to highlight the relationship between classes “CProblem” and “CConstraint”, which expresses that

a problem will have lists of constraints associated, one of hard constraints and another of soft constraints. A penalty is associated with each soft constraint in this list, symbolizing how much the quality of the solution is affected by a violation of such a constraint.

In this way, once constraints for solving a problem are implemented, it can be easily configured to meet other problems of the same class. For example, an implementation performed to solve an UCTTP of a specific institution can be simply tuned to another institution. This can be done by: (i) removing or inserting constraints in one of the lists (restrictions that exist for one problem may not exist for another); (ii) changing the constraints from one list to another (hard constraints for one problem may be soft constraints for another, and vice versa); and (iii) changing the penalty associated with soft restrictions (violation of a constraint may be more restrictive from one problem to another).

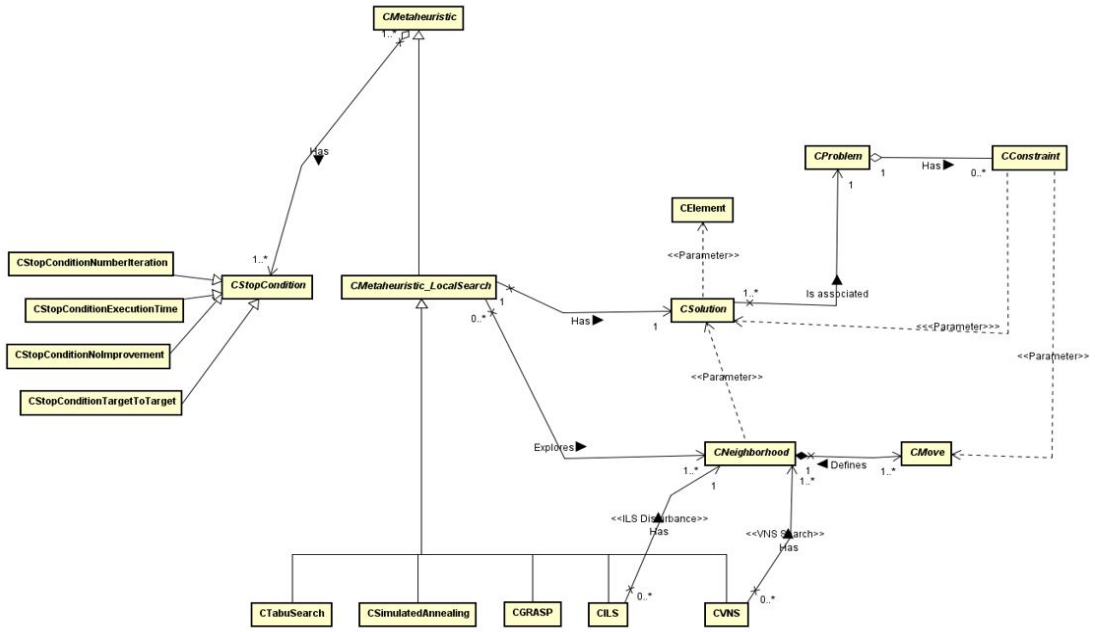


Figure 3. Class diagram of framework FINESS (Vianna et al., 2014).

Analytic Hierarchy Process

Analytic Hierarchy Process (AHP) was presented in (Saaty, 1980; 2001) as a technique for choosing one among a set of alternatives, considering a set of criteria (objectives). In a macro way, AHP has two main phases: (i) definition of weight (preferences) of each criterion; and (ii) definition of priority of each alternative according to the criteria involved. In (i), a pairwise comparison between the criteria is performed to define the weight of each criterion. In (ii), for each criterion, a pairwise comparison between the alternatives is performed to define the priority of each one.

In the scientific literature, it is common to find articles that use the method AHP to: (i) select one of a set of alternatives (Santis et al., 2017; Qin et al.; 2019; Tavana et al., 2019); and (ii) classify a specific scenario into a set of predefined categories (Chakraborty et al., 2019; Karan et al., 2019; Yang et al., 2019). However, some articles use AHP to define weights, which will serve as input to other methods, such as fuzzy logic (Agápito et al., 2019; Peng et al., 2019), heuristics (Wang et al., 2018; Toktas and Can, 2019), etc.

According to Saaty (1980; 2001), each element a_{ij} of a pairwise comparison matrix (Matrix *A*) is defined as Equations 1 and 2.

$a_{ij} = 1$ for $i=j$ (1)

$$a_{ij} = \frac{1}{a_{ij}} \text{ for } i \neq j,\ 1 \leq i, j \leq n \quad (2)$$

where n is the number of criteria or alternatives under analysis.

Judgement scale used in this work is the same one proposed by Saaty (1980), which is described in Chart 1.

Chart 1. Scale for pairwise comparisons

Relative intensity	Definition	Explanation
1	Of equal value	Two requirements are of equal value
3	Slightly more value	Experience slightly favors on requirement over another
5	Essential or strong value	Experience strongly favors on requirement over another
7	Very strong value	A requirement is strongly favored and its dominance is demonstrated in practice
9	Extreme value	Evidence favoring one over another is of the highest possible order of affirmation
2,4,6,8	Intermediate values between two adjacent judgments	When compromise is needed

Adapted from Saaty (1980).

According to Saaty's method (Saaty, 1980), the preference matrix A should be normalized, generating the matrix B (Equations 3 and 4).

$$B = \left[b_{ij} \right] \quad (3)$$

$$b_{ij} = \frac{a_{ij}}{\sum_{k=1}^{n} a_{kj}} \quad (4)$$

In order to calculate the eigenvector $w=[w_i]$, which expresses the preference (weight) of the elements, the Equation 5 is used.

$$w_i = \frac{\sum_{j=1}^{n} b_{ij}}{n} \quad (5)$$

The next step involves the consistency check (Saaty, 1980; 2001) of comparisons by calculating the maximal eigenvalue, according to the Equations 6 and 7.

$$\lambda_{max} = \frac{1}{n} \sum_{i=1}^{n} \frac{(Aw)_i}{w_i} \quad (6)$$

$$CI = \frac{\lambda_{max} - n}{n - 1} \quad (7)$$

where λ_{max} is the matrix's largest eigenvalue and CI is the consistency index.

Using the random consistency index table from Saaty (1980), we can calculate the consistency ratio *CR* with the Equation 8.

$$CR = \frac{CI}{RI}.100\% \quad (8)$$

If *CR* is less than 10%, the pairwise comparison matrix is consistent (Saaty, 1980; 2001).

3. METHODOLOGY

To achieve the objectives described in Section "Introduction", the following methodology was followed:

- Data acquisition: data were collected from the coordinators of the undergraduate courses in Computer Science and Industrial Engineering at the Fluminense Federal University (UFF);
- Problem solving: Hybrid heuristics, which combine metaheuristics VNS and Tabu Search, have been proposed. These heuristics were developed with the aid of the FINESS framework (described in Section "Framework FINESS");
- Experiments: they were conducted over four real instances, referring to the second semester of 2016 and the first semester of 2017 of the undergraduate courses in Computer Science and Industrial Engineering at UFF;
- Validation of results: The results obtained by the proposed hybrid heuristics were compared with the manual solution obtained by the course coordinators. The results were also compared with standard versions of the metaheuristics VNS (algorithm of Figure 1) and Tabu Search (algorithm of Figure 2), which were also developed with the aid of the FINESS framework.

4. SOLUTION APPROACHES

Four heuristics were developed using the FINESS framework for granting fast development. For a better understanding, objective function and solution encoding is presented in Section "Objective function and solution encoding", method used to set soft constraints violation penalties is described in Section "Definition of violations penalties", constructive algorithm is presented in Section "Constructive method", neighborhood structures used are described in Section "Neighborhood structures", the chosen local search method is presented in Section "Local search method" and finally proposed heuristics are detailed in Section "Proposed heuristics".

Objective function and solution encoding

Let p_i be the penalty related to the violation of soft constraint *i* and $v_i(s)$ the amount of violations of constraint *i* in solution *s*. The objective function $f(s)$ is given by the Equation 9.

$$f(s) = \sum p_i v_i(s) \quad (9)$$

p_i values were obtained using a multicriteria decision analysis method, which is detailed in Section "Definition of violations penalties".

A solution to the addressed problem was encoded as a three-dimensional matrix $M_{q \times m \times n}$, where *m*, *n* and *q* represent, respectively, the number of timeslots, the number of days and the number of semester groups. Each position (*i*,*j*,*l*) of the matrix *M* has the class, of semester group *l*, allocated at timeslot *i* of day *j*. if *Mi*(*i*,*j*,*l*)=0, there is no class allocated.

Definition of violations penalties

As described in the previous section, quality of solution is associated with the amount of soft constraints violated. Less violation implies better solutions.

As already mentioned, there are three groups of soft constraints. Constraints of Group 3 when violated turn solution almost unfeasible. Therefore, a high penalty to constraints of this group are applied. Penalty was set at 10000, which was chosen so that this group stands out from the other two.

The penalties of the other two groups were defined using multicriteria decision analysis. However, due to the importance of each group, the range of values in which each group is comprised is defined a priori. Penalties of constraints of Group 1 were defined as between 1 and 10. The penalties of the constraints of Group 2 were defined as between 100 and 500.

The multicriteria decision method used was "Analytic Hierarchy Process" (AHP), which defines a preference (between 0 and 1) for each of the criteria analyzed. After applying the AHP method, preferences are scaled to the range defined for each group of constraints, generating penalty values.

The application of the method AHP for defining the violation penalties of the constraints of Group 1 and 2 will be described in Section 5.

Constructive method

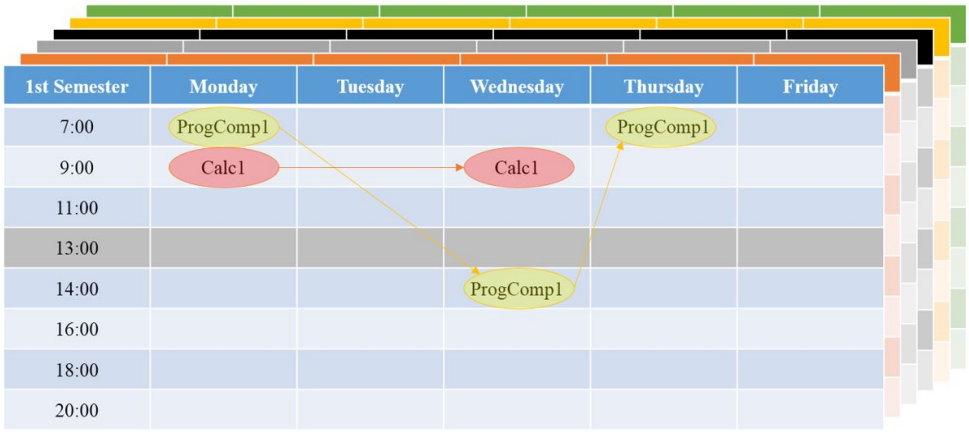
In the proposed constructive method, classes are allocated according to the following preference order: (i) classes assigned to professors with many time restrictions; (ii) course shift preference; and (iii) timeslots in chosen shift are considered.

When inserting a class *x* in the partial solution, *x* is divided into 2-hours allocations for which priority is given to the insertion in the same shift (morning, afternoon or evening), at the same time and on nonconsecutive days.

Figure 4 depicts a class scheduling example, in which two classes are allocated: Calc1 and ProgComp1. Class Calc1 has 4 hours (Mondays and Wednesdays at 9 a.m.) and cannot be scheduled (it has a fixed schedule). Class ProgComp1 has 6 hours and his professor is unavailable on Wednesday mornings and on Fridays. Allocation of classes prioritizes morning period, then afternoon period and finally evening period. As Calc1 has a fixed schedule, it is allocated first, at 9 a.m. on Mondays and Wednesdays. Then ProgComp1 is allocated at 7 a.m. on Monday. As his professor is unavailable at Wednesday mornings, ProgComp1 is allocated at 2 p.m. on Wednesday. Finally, since his professor is unavailable on Fridays, ProgComp1 is allocated at 7 a.m. on Thursdays.

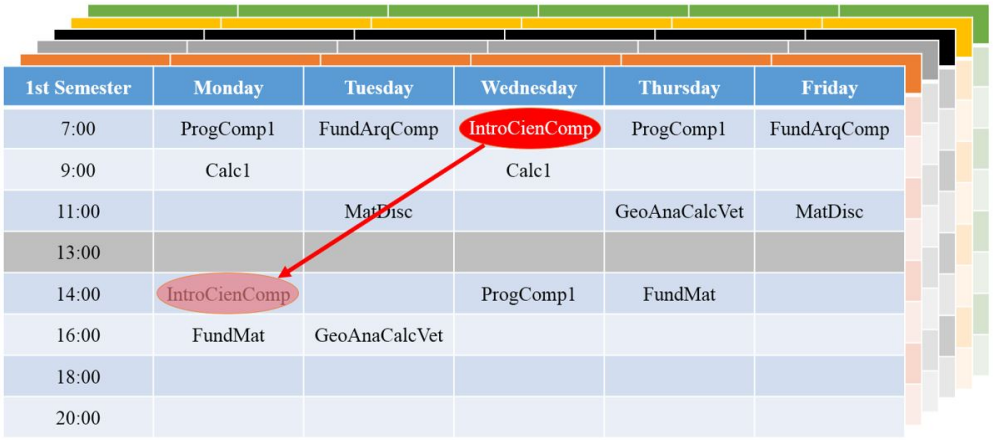
Neighborhood structures

Two neighborhood structures were used: *k*-Reschedule and *k*-Exchange. Movements in those neighborhood structures happens inside a semester group. In *k*-Reschedule neighborhood, *k* allocations are randomly chosen and reschedule, i.e. each of the *k* allocations is removed from the current day/time and inserted at another day/time. Figure 5 exemplifies 1-Reschedule. Class "IntroCienComp" previously allocated on Wednesday at 7 a.m. is moved to Monday at 2 p.m.



1st Semester	Monday	Tuesday	Wednesday	Thursday	Friday
7:00	ProgComp1			ProgComp1	
9:00	Calc1		Calc1		
11:00					
13:00					
14:00			ProgComp1		
16:00					
18:00					
20:00					

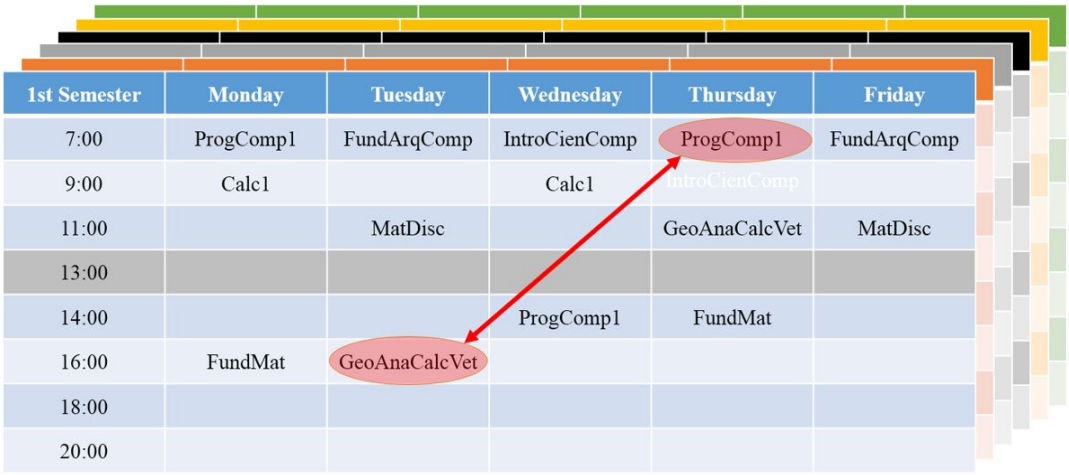
Figure 4. Constructive method example.



1st Semester	Monday	Tuesday	Wednesday	Thursday	Friday
7:00	ProgComp1	FundArqComp	IntroCienComp	ProgComp1	FundArqComp
9:00	Calc1		Calc1		
11:00		MatDisc		GeoAnaCalcVet	MatDisc
13:00					
14:00	IntroCienComp		ProgComp1	FundMat	
16:00	FundMat	GeoAnaCalcVet			
18:00					
20:00					

Figure 5. Example of 1-Reschedule move.

In *k*-Exchange neighborhood, *k* allocation pairs (a_1, a_2) are randomly chosen. Each allocation pair is inverted, i.e. allocations a_1 and a_2 have their positions exchanged. Figure 6 exemplifies a 1-Exchange movement between classes “GeoAnaCalcVet” and “ProgComp1”.



1st Semester	Monday	Tuesday	Wednesday	Thursday	Friday
7:00	ProgComp1	FundArqComp	IntroCienComp	ProgComp1	FundArqComp
9:00	Calc1		Calc1	IntroCienComp	
11:00		MatDisc		GeoAnaCalcVet	MatDisc
13:00					
14:00			ProgComp1	FundMat	
16:00	FundMat	GeoAnaCalcVet			
18:00					
20:00					

Figure 6. Example of 1-Exchange move.

Local search method

This work uses method VND (Variable Neighborhood Descent) (Mladenovic and Hansen, 1997) as local search strategy since it allows multiple neighborhood structures. Figure 7 presents VND algorithm, which receives as input parameters the solution s_0 to be refined and the neighborhood structures N_1=1-Exchange, N_2=2-Reschedule and N_3=2-Exchange.

```
Procedure VND (s0, N1, N2, N3)
1.  Let s be the best current solution. s ← s0;
2.  k ← 1;
3.  While k ≤ 3 do
4.    Apply a local search in s using neighborhood Nk. Let s' be the local optimum;
5.    If f(s') < f(s) then
6.      s ← s';
7.      k ← 1;
8.    Else
9.      k ← k + 1;
10.   End-if
11. End-while
12. Return s;
```

Figure 7. VND algorithm.

Proposed heuristics

In this work, four heuristics were proposed:

1. **VNS** - follows the VNS framework described in Figure 1. In the VNS disturbance procedure, the neighborhoods 3-Reschedule and 3-Exchange were used.
2. **TS** - follows the Tabu Search framework described in Figure 2. As local search procedure, a method VND-TS was used, which is based on the VND algorithm described in Figure 7, using tabu search to investigate the neighborhood N_k (Line 4). The tabu search procedure is interrupted when a solution better than s (current best solution) is found or when 200 iterations without improvement are performed. For each neighborhood, a different tabu tenure was defined: 100 for 1-Exchange and 2-Exchange; and 50 for 2-Reschedule. This VND-TS method is different from that used by Cruz et al. (2012) and Zeng et al. (2016), since the TS method is not used as an intensification process applied after the execution of the VND search.
3. **VNS-TS1** - adapts the VNS framework described in Figure 1, replacing the local search procedure by the Tabu Search method. In the VNS disturbance procedure, the neighborhoods 3-Reschedule and 3-Exchange were used. As local search procedure, method VND-TS described at Item 2 was used.
4. **VNS-TS2** - follows the VNS framework described in Figure 1, inserting an intensification phase, which is performed every 50 VNS iterations. During these 50 iterations, a pool with the 5 best solutions is defined which is investigated, in the intensification phase, by the VND-TS method described at Item 2. In the VNS disturbance procedure, the neighborhoods 3-Reschedule and 3-Exchange were used.

The values of the input parameters and the neighborhood structures used by the proposed heuristics were defined by experiments.

5. RESULTS AND DISCUSSION

The experiments were conducted using a 1.10GHz Intel Processor. Initially, the violation penalties of the soft constraints of Groups 1 and 2 were set. Then, the proposed heuristics were evaluated.

Definition of penalties of the constraints of Groups 1 and 2

As already reported, AHP has two main phases: (i) definition of weight (preferences) of each criterion; and (ii) definition of priority of each alternative according to the criteria involved. In this study, only the first phase will be held.

Initially, we will define penalties of Group 1 constraints, which is formed by the following constraints (already defined in Section "Addressed problem"):

- R1.1 - Student classes collision;
- R1.2 - Professor maximum days – refers to the least strong case of this constraint in which, if the maximum number of days is violated the quality of the solution is decreased, but it is still feasible.
- R1.3 - Class maximum hours per day – refers to the case that if the maximum number of hours is violated the quality of the solution is decreased, but it is still feasible;
- R1.4 - Class allocated time.

Each of these restrictions were treated as a criterion, for which a weight should be defined by AHP method. Each criterion was pairwise compared with others and placed in the preference matrix *A* described in Table 1. For such comparison, a set of specialists (current and former coordinators of undergraduate courses) were gathered in a room, which reached a consensus.

Table 1. Preference matrix *A* of Group 1 of constraints

	R1.1	**R1.2**	**R1.3**	**R1.4**
R1.1	1	3	6	9
R1.2	1/3	1	4	7
R1.3	1/6	1/4	1	5
R1.4	1/9	1/7	1/5	1

The normalized matrix *B* is presented in Table 2 - the last column presents the eigenvector *w*.

Table 2. Normalized comparison matrix *B* of Group 1 of constraints

	R1.1	**R1.2**	**R1.3**	**R1.4**	***w***
R1.1	0.62	0.68	0.54	0.41	0.57
R1.2	0.21	0.23	0.36	0.32	0.28
R1.3	0.10	0.06	0.09	0.23	0.10
R1.4	0.07	0.03	0.02	0.05	0.04

Consistency ratio found was *CR* = 7.777%, which is consistent because it is less than 10% (Saaty, 1980; 2001).

To scale the weights (eigenvector *w*) found between 1 and 10 (range defined for Group 1), the Equation 10 was used.

$$p_i = \frac{(w_i - w_{min})(10-1)}{w_{max} - w_{min}} + 1 \quad (10)$$

where w_{max} and w_{min} represent respectively the largest and smallest values of w and p_i represents the penalty associated to constraint i. Penalties associated to constraints R1.1, R1.2, R1.3 and R1.4 were respectively set in 10, 6, 3 and 1.

To obtain Group 2 constraints penalties, the same procedure was followed. Group 2 is formed by the following constraints (already defined in Section "Addressed problem"):

- R2.1 - Class allocated day;
- R2.2 - Class available time period;
- R2.3 - Professor unavailable time – refers to the case to available times that, if possible, the allocation of classes should be avoided;
- R2.4 - Professor time windows;
- R2.5 - Class maximum hours per day – refers to the case that the maximum number of hours should be considered at most. However, if this number is violated the quality of the solution is greatly decreased, but it is still feasible;
- R2.6 - Class time period.

The preference matrix A is described in Table 3. For such pairwise comparison, a set of specialists (current and former coordinators of undergraduate courses) were gathered in a room, which reached a consensus.

Table 3. Preference matrix A of Group 3 of constraints

	R2.1	**R2.2**	**R2.3**	**R2.4**	**R2.5**	**R2.6**
R2.1	1	4	1/9	3	2	2
R2.2	1/4	1	1/9	1/3	1/4	1/4
R2.3	9	9	1	9	9	9
R2.4	1/3	3	1/9	1	1/2	1/2
R2.5	1/2	4	1/9	2	1	1
R2.6	1/2	4	1/9	2	1	1

Normalized matrix B is presented in Table 4 - last column presents eigenvector w.

Table 4. Normalized comparison matrix B of Group 2 of constraints

	R2.1	**R2.2**	**R2.3**	**R2.4**	**R2.5**	**R2.6**	***w***
R2.1	0.09	0.16	0.07	0.17	0.15	0.15	0.13
R2.2	0.02	0.04	0.07	0.02	0.02	0.02	0.03
R2.3	0.78	0.36	0.64	0.52	0.65	0.65	0.61
R2.4	0.03	0.12	0.07	0.06	0.04	0.04	0.05
R2.5	0.04	0.16	0.07	0.12	0.07	0.07	0.09
R2.6	0.04	0.16	0.07	0.12	0.07	0.07	0.09

Consistency ratio found was CR = 7.800%, which is consistent because it is less than 10% (Saaty, 1980; 2001).

To scale the weights (eigenvector w) found between 100 and 500 (range defined for Group 2), the Equation 11 was used.

$$p_i = \frac{(w_i - w_{min})(500 - 100)}{w_{max} - w_{min}} + 100 \quad (11)$$

where w_{max} and w_{min} are respectively the largest and smallest values of w and p_i is the penalty associated to constraint i. Penalties associated to constraints R2.1, R2.2, R2.3, R2.4, R2.5 and R2.6 were respectively set in 170, 100, 500, 118, 140 and 140.

Evaluation of the proposed heuristics

The proposed heuristics were coded in C++, using the Visual Studio 2018 compiler. The performance of the proposed heuristics was evaluated on a set of 4 real instances, which were approached by the coordination of the undergraduate courses in Computer Science and Industrial Engineering of Fluminense Federal University (UFF) in the second semester of 2016 and in the first semester of 2017.

Each proposed heuristic was executed five times, in which each execution lasted 300 seconds.

Chart 2 shows the instances used at the computational experiments. For each instance is presented: name, number of classes (*c*), number of professors (*p*) and the cost of the solution manually obtained by the undergraduate course coordination.

Table 5 and 6 present the results. Table 5 shows, for each instance and for each heuristic, the cost of the best solution (*Best*) found and the *Gap* - percentage of improvement of the best solution cost in relation to the solution manually obtained. The best results were highlighted in bold.

Chart 2. Instances description

Name	c	*p*	Manual
C16	57	26	1088
C17	59	26	1388
E16	71	33	1830
E17	74	33	2112

Table 6 shows, for each instance and for each heuristic, the average cost (*Avg*) obtained in five executions and the *Gap* - percentage of improvement of the average cost (*Avg*) in relation to the solution manually obtained. Best results were highlighted in bold.

Table 5. Best solution costs obtained by each proposed heuristic

Instance		VNS		TS		VNS-TS1		VNS-TS2	
Name	Manual	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)
C16	1088	**955**	12.2	970	10.8	**955**	12.2	**955**	12.2
C17	1388	1192	14.1	1202	13.4	**1190**	14.3	**1190**	14.3
E16	1830	1646	10.1	1651	9.8	1638	10.5	**1634**	10.7
E17	2112	1782	15.6	1784	15.5	**1775**	16.0	1782	15.6

Table 6. Average solution costs obtained by each proposed heuristic

Instance		VNS		TS		VNS-TS1		VNS-TS2	
Name	Manual	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)	Cost	Gap (%)
C16	1088	964.8	11.3	980.2	9.9%	**957.0**	12.0	961.0	11.7
C17	1388	1198.3	13.7	1211.0	12.7%	**1194.3**	13.9	1196.2	13.8
E16	1830	1650.7	9.8	1657.2	9.4%	**1643.2**	10.2	1644.5	10.1

Table 6. Continued...

Instance		VNS		TS		VNS-TS1		VNS-TS2	
Name	**Manual**	**Cost**	**Gap (%)**	**Cost**	**Gap (%)**	**Cost**	**Gap (%)**	**Cost**	**Gap (%)**
E17	2112	1797.7	14.9	1802.3	14.7%	**1783.9**	15.5	1792.3	15.1

All proposed heuristics outperform manual solutions. Hybrid heuristics obtained the best results, with emphasis on the heuristic VNS-TS1. Analyzing the average cost obtained on five executions, VNS-TS1 obtained best results for all instances. When the best solutions found are compared, VNS-TS1 obtained the best results in three out of four instances, being surpassed by the heuristic VNS-TS2 on the other instance. VNS-TS2 obtained the best solution for three out of four instances.

6. FINAL CONSIDERATIONS AND RECOMMENDATIONS

In this work, four heuristics were developed for the university course timetabling problem (UCTTP), which were applied in the Computer Science and Industrial Engineering undergraduate courses of Fluminense Federal University. The heuristics were developed with the support of the FINESS framework, which, among other virtues, allows the heuristic extensibility, that is, the process of adding and removing constraints is simple, as well as changing the values of penalties of soft constraints. This feature allows proposed heuristics to be more easily adaptable to UCTTP of other educational institutions.

The computational results showed the efficiency of the hybrid VNS-TS heuristics, with emphasis on the heuristic VNS-TS1, which uses tabu search as local search method. VNS-TS1 obtained best average results for all instances and found best solution for three out of four instances. The percentage of improvement obtained by this heuristic, in relation to the manual solution, ranged in average between 12.0 to 15.5%. The heuristic VNS-TS2, which uses tabu search as an intensification strategy, also achieved good results.

In addition to the heuristics, it is also proposed, as an original contribution of this work, a method VND-TS which combines VND flexibility with deep investigation of Tabu Search. This method is used as local search phase of heuristic VNS-TS1, which contributed to the success of this heuristic.

The proposed heuristic VNS-TS1 has been in use since the first half of 2019 by the coordination of the undergraduate courses in Computer Science and Industrial Engineering of Fluminense Federal University (UFF). Even though each university has specific constraints on its timetable, the use of the proposed heuristic is possible because it has been developed using a framework that allows for the easy insertion and removal of constraints. In other words, little programming effort will be needed to adapt the proposed heuristic to another university timetabling problem.

The next step of this study is the development of a collaborative computer system for the university time allocation problem. This system will make use of collaborative techniques that will gather “suggestions” from teachers and students in the elaboration of the timetable. The heuristic VNS-TS1 will be used by the collaborative system to solve the timetabling problem.

REFERENCES

Abdelhalim, E.A. and El Khayat, G.A. (2016), "A utilization-based genetic algorithm for solving the university timetabling problem (UGA)", *Alexandria Engineering Journal*, Vol. 55, No. 2, pp. 1395-409.

Abdullah, S., Turabieh, H., McCollum, B., McMullan, P. (2012), "A hybrid metaheuristic approach to the university course timetabling problem", *Journal of Heuristics*, Vol. 18, No. 1, pp. 1-23.

Agápito, A.O., Vianna, M.F.D., Moratori, P.B. et al. (2019), "Using multicriteria analysis and fuzzy logic for project portfolio management", *Brazilian Journal of Operations & Production Management*, Vol. 16, No. 2, pp. 347-57.

Al-Betar, M.A. and Khader, A.T. (2012), "A harmony search algorithm for university course timetabling", *Annals of Operations Research*, Vol. 194, No. 1, pp. 3-31.

Arenas, M.G., Dolin, B., Merelo, J.J. et al. (2002), "JEO: Java evolving objects", in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Springer, Berlin, pp. 991-4.

Asratian, A.S. and Werra, D. (2002), "A generalized class-teacher model for some timetabling problems", *European Journal of Operational Research*, Vol. 143, pp. 531-42.

Avella, P. and Vasil'ev, I. (2005), "A computational study for a cutting plane algorithm for university course timetabling", *Journal of Scheduling*, Vol. 8, pp. 497-514.

Babaei, H., Karimpour, J. and Hadidi, A. (2015), "A survey of approaches for university course timetabling problem", *Computers & Industrial Engineering*, Vol. 86, pp. 43-59.

Bellio, R., Ceschia, S., Di Gaspero, L. et al. (2016), "Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem", *Computers & Operations Research*, Vol. 65, pp. 83-92.

Birbas, T., Daskalaki, S. and Housos, E. (2009), "School timetabling for quality student and teacher schedules", *Journal of Scheduling*, Vol. 12, pp. 177-97.

Bolaji, A.L., Khader, A.T., Al-Betar, M.A. et al. (2014), "University course timetabling using hybridized artificial bee colony with hill climbing optimizer", *Journal of Computational Science*, Vol. 5, No. 5, pp. 809-18.

Borchani, R., Abdelkerim, E. and Masmoudi, M. (2017), "Variable neighborhood descent search based algorithms for course timetabling problem: Application to a Tunisian University", *Electronic Notes in Discrete Mathematics*, Vol. 58, pp. 119-126.

Burke, E.K., Eckersley, A.J., McCollum, B. et al. (2010), "Hybrid variable neighborhood approaches to university exam timetabling", *European Journal of Operational Research*, Vol. 206, No. 1, pp. 46-53.

Cahon, S., Melab, N. and Talbi, E. (2004), "ParadisEO: a framework for the reusable design of parallel and distributed metaheuristics", *Journal of Heuristics*, Vol. 10, pp. 357-80.

Cai, Y. and Zhu, D. (2019), "Trustworthy and profit: A new value-based neighbor selection method in recommender systems under shilling attacks", *Decision Support Systems*, Vol. 124, pp. 113112.

Chakraborty, S., Sahoo, S., Majumdar, D. et al. (2019), "Future Mangrove Suitability Assessment of Andaman to strengthen sustainable development", *Journal of Cleaner Production*, Vol. 234, pp. 597-614.

Cruz, R.C., Silva, T.C.B., Souza, M.J.F. et al. (2012), "GENVNS-TS-CL-PR: a heuristic approach for solving the vehicle routing problem with simultaneous pickup and delivery", *Electronic Notes in Discrete Mathematics*, Vol. 39, pp. 217-24.

Delgoshaei, A., Mirzazadeh, A. and Ali, A. (2018), "A Hybrid Ant Colony System and Tabu Search algorithm for the production planning of dynamic cellular manufacturing systems while confronting uncertain costs", *Brazilian Journal of Operations & Production Management*, Vol. 15, No. 4, pp. 499-516.

Di Gaspero, L. and Schaerf, A. (2003), "EASYLOCAL++: an object-oriented framework for flexible design of local search algorithms", *Software, Practice & Experience*, Vol. 33, No. 8, pp. 733.

Feng, X., Lee, Y. and Moon, I. (2017), "An integer program and a hybrid genetic algorithm for the university timetabling problem", *Optimization Methods & Software*, Vol. 32, No. 3, pp. 625-49.

Fink, A. and Voß, S. (2002), "Hotframe: a heuristic optimization framework in optimization software class libraries", in Voß, S. and Woodruff, D.L. (Eds.), *Optimization Software Class Libraries*, Springer US, Boston, pp. 81-154.

Glover, F. (1989), "Tabu search, part I", *ORSA Journal on Computing*, Vol. 1, No. 3, pp. 190-206.

Glover, F. (1990), "Tabu search, part II", *ORSA Journal on Computing*, Vol. 2, No. 1, pp. 5-32.

Gülcü, A. and Bulkan, S. (2019), "Integer programming versus constraint programming: a course timetabling case study", *International Journal of Industrial Engineering: Theory Applications and Practice*, Vol. 26, No. 3, pp. 301-16.

Jat, S.N. and Yang, S.A. (2011), "Hybrid genetic algorithm and tabu search approach for post enrollment course timetabling", *Journal of Scheduling*, Vol. 14, No. 6, pp. 617-37.

Kahar, M.N.M. and Kendall, G. (2010), "The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution", *European Journal of Operational Research*, Vol. 207, pp. 557-65.

Karan, S.K., Ghosh, S. and Samadder, S.R. (2019), "Identification of spatially distributed hotspots for soil loss and erosion potential in mining areas of Upper Damodar Basin - India", *Catena*, Vol. 182, pp. 104144.

Keijzer, M., Merelo, J.J., Romero, G. et al. (2001), "Evolving objects: general purpose evolutionary computation library", in *Proceedings of the 5th International Conference on Artificial Evolution (EA 2001)*, Springer, Switzerland, pp. 231-42.

Lau, H.C., Wan, W.C., Halim, S. et al. (2007), "A software framework for fast prototyping of meta-heuristics hybridization", *International Transactions in Operational Research*, Vol. 14, No. 2, pp. 123-41.

Lewis, R. (2008), "A survey of metaheuristic-based techniques for University Timetabling problems", *OR-Spektrum*, Vol. 30, No. 1, pp. 167-90.

Lima, T.J.B. (2016), *Colaborário: Sistema Colaborativo para Geração de Quadro de Horários de Curso Universitário*, Dissertação de Mestrado em Engenharia de Produção e Sistemas Computacionais, Universidade Federal Fluminense, Rio das Ostras, RJ.

Lindahl, M., Sørensen, M. and Stidsen, T.R. (2018), "A fix-and-optimize matheuristic for university timetabling", *Journal of Heuristics*, Vol. 24, No. 4, pp. 645-65.

Lohpetch, D. and Jaengchuea, S. (2016), "A hybrid multi-objective genetic algorithm with a new local search approach for solving the post enrolment based course timetabling problem", *Advances in Intelligent Systems and Computing*, Vol. 463, pp. 195-206.

Lukasiewycz, M., Glaß, M., Reimann, F. et al. (2011), "Opt4J: a modular framework for meta-heuristic optimization", in *Proceedings of the 13th Conference on Genetic and Evolutionary Computation (GECCO)*, ACM, New York, pp. 1723-30.

Mazlan, M., Makhtar, M., Khairi, A.F.K.A. et al. (2019), "University course timetabling model using ant colony optimization algorithm approach", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 13, No. 1, pp. 72-6.

Mladenovic, N. and Hansen, P. (1997), "Variable neighborhood search", *Computers & Operations Research*, Vol. 24, pp. 1097-100.

Peng, T., Li, C. and Zhou, X. (2019), "Application of machine learning to laboratory safety management assessment", *Safety Science*, Vol. 120, pp. 263-7.

Pereira, V. and Costa, H.G. (2016), "Linear integer model for the course timetabling problem of a faculty in Rio de Janeiro", *Advances in Operations Research*, Vol. 2016, pp. 1-9.

Phillips, A.E., Walker, C.G., Ehrgott, M. et al. (2017), "Integer programming for minimal perturbation problems in university course timetabling", *Annals of Operations Research*, Vol. 252, No. 2, pp. 283-304.

Pillay, N. and Özcan, E. (2019), "Automated generation of constructive ordering heuristics for educational timetabling", *Annals of Operations Research*, Vol. 275, No. 1, pp. 181-208.

Prabodanie, R.A.R. (2017), "An integer programming model for a complex university timetabling problem: a case study", *Industrial Engineering and Management Systems*, Vol. 16, No. 1, pp. 141-53.

Qin, C., Li, B., Shi, B. et al. (2019), "Location of substation in similar candidates using comprehensive evaluation method base on DHGF", *Measurement: Journal of the International Measurement Confederation*, Vol. 146, pp. 152-8.

Qu, R. and Burke, E.K. (2009), "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems", *The Journal of the Operational Research Society*, Vol. 60, No. 9, pp. 1273-85.

Saaty, T. (1980), *The Analytic Hierarchy Process*, McGraw-Hill, New York.

Saaty, T. (2001), *Decision Making for Leaders: the Analytic Hierarchy Process for Decisions in a Complex Word*, University of Pittsburgh, Pittsburgh.

Santis, R.B., Golliat, L. and Aguiar, E.P. (2017), "Multi-criteria supplier selection using fuzzy analytic hierarchy process: case study from a Brazilian railway operator", *Brazilian Journal of Operations & Production Management*, Vol. 14, No. 3, pp. 428-37.

Schaerf, A. (1999), "A survey of automated timetabling", *Artificial Intelligence Review*, Vol. 13, No. 2, pp. 87-127.

Schermer, D., Moeini, M. and Wendt, O. (2019), "A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations", *Computers & Operations Research*, Vol. 109, pp. 134-58.

Socha, K., Sampels, M. and Manfrin, M. (2003), “Ant algorithms for the university course timetabling problem with regard to the state-of-the-art”, in *EvoWorkshops 2003: Applications of Evolutionary Computing*, Springer-Verlag, Berlin, pp. 334–45.

Tavana, M., Khosrojerdi, G., Mina, H. et al. (2019), "A hybrid mathematical programming model for optimal project portfolio selection using fuzzy inference system and analytic hierarchy process", *Evaluation and Program Planning*, Vol. 77, pp. 101703.

Toktaş, P. and Can, G.F. (2019), "Stochastic KEMIRA-M approach with consistent weightings", *International Journal of Information Technology & Decision Making*, Vol. 18, No. 3, pp. 793-831.

Troiano, L. and Pasquale, D.D. (2010), “Java library for genetic algorithms addressing memory and time issues”, in *Second World Congress on Nature and Biologically Inspired Computing (NaBIC)*, IEEE, Coimbatore, India, available at: https://ieeexplore.ieee.org/document/5393443 (accessed 14 December 2018).

Ventura, S., Romero, C., Zafra, A. et al. (2008), "JCLEC: a Java framework for evolutionary computation", *Soft Computing*, Vol. 12, No. 4, pp. 381-92.

Vianna, D.S., Martins, C.B., Medeiros, A.P. et al. (2014), *FINESS: Framework for the Implementation of Metaheuristics Based on Neighborhood Structure Search*, Technical report, Federal Fluminense University, Niterói.

Voudouris, C., Dorne, R., Lesaint, D. et al. (2001), “iOpt: a software toolkit for heuristic search methods in principles and practice of constraint programming”, in Walsh, T. (Ed.), *Principles and Practice of Constraint Programming: CP 2001*, Springer-Verlag, Berlin, pp. 716-29.

Wagner, S. and Affenzeller, M. (2005), “HeuristicLab: a generic and extensible optimization environment in adaptive and natural computing algorithms”, in Ribeiro, B., Albrecht, R.F., Dobnikar, A. et al. (Eds.), *Adaptive and Natural Computing Algorithms: Proceedings of the International Conference*, Springer, Vienna, pp. 538-41.

Wang, H., Zhang, W. and Liu, Y. (2018), "A robust measurement placement method for active distribution system state estimation considering network reconfiguration", *IEEE Transactions on Smart Grid*, Vol. 9, No. 3, pp. 2108-17.

Werra, D. (1985), "An introduction to timetabling", *European Journal of Operational Research*, Vol. 19, No. 2, pp. 151-62.

Yang, Z., Li, W., Li, X. et al. (2019), "Assessment of eco-geo-environment quality using multivariate data: a case study in a coal mining area of Western China", *Ecological Indicators*, Vol. 107, pp. 105651.

Zeng, Z., Yu, X., He, K. et al. (2016), "Iterated Tabu Search and Variable Neighborhood Descent for packing unequal circles into a circular container", *European Journal of Operational Research*, Vol. 250, No. 2, pp. 615-27.

Authors contribution: All the authors contributed equally to this paper.