International Journal of Engineering Research and Technology. ISSN 0974-3154, Volume 14, Number 3 (2021), pp. 241-248

Class schedules assignment in a school in Colombia through Boolean satisfiability

A. Ramirez-Restrepo[1], Y.F. Ceballos[2*] and L.E. Muñoz[3]

[1,2] Grupo Ing. y sociedad. Ingeniería Industrial, Universidad de Antioquia Calle 70 No 52-21, Medellín 050010, Colombia.
[3]Universidad Tecnológica de Pereira.Pereira, Colombia.
**Corresponding Author (ORCID: 0000-0001-5787-8832)*

Abstract:

Introduction: NP-Complete sentences are difficult problems to find a solution since they are part of the set of decision problems that can be solved in polynomial time. Some of them, such as the Boolean satisfiability problem and the traveling salesman problem, have been the protagonists of several resolution methods, among them graph coloring, simulated travel, genetic algorithms, among others. The allocation of schedules is classified as one of them, which consists of assigning a series of subjects to daily work schedules in an educational institution or in a company. Although in some educational institutions this work is easy to do, it is a problem that involves programming and analysis skills. This is since generally there is very little time to make a feasible schedule, as well as other important factors such as the configuration of the days, the subjects, the restrictions of teachers and spaces; the professors whose subjects must be taught according to the academic profile of each one; the classrooms where each teacher must work and what subjects are related, among others. **Objective**: propose an adjusted method for the problem of assigning class schedules based on Graph Colorization and Boolean Satisfaction (SAT) to be applied in an educational institution in Colombia. **Methodology**: The Visual Basic 6.0 programming language was used as a tool, a powerful programming system that allows to build applications quickly and efficiently for Microsoft Windows. **Results**: The experiments carried out show that, in all the test cases, the algorithm based on graph coloring and Boolean satisfiability SAT finds better solutions than those obtained with the manual method, in a reasonable computational time. **Conclusions**: The generated model allows obtaining schedules for each teacher and each group of students. Although the problem was modeled and solved specifically for just one educational institution, the model allows adjusting to many scheduling problems in different educational institutions and other workspaces, providing great flexibility in resolution.

Keywords: Algorithm; Graph Coloring, Educational Institution, NP-Complete; Boolean Compliance Programming; Software

I. INTRODUCTION

The problem of assigning class schedules consists of assigning a class series daily for work schedules in an educational institution and belongs to the NP-Complete problems. NP-Complete problems are difficult to solve and are part of the NP complexity class, which is the set of decision problems that can be solved by a non-deterministic machine in polynomial time [1], [2]. This class contains many problems that you usually want to solve in practice, including the Boolean satisfiability problem and the traveling salesman problem, a Hamiltonian path to go through all vertices only once [3], [4]. All problems of this class have the property that their solution can be effectively verified. This problem has been studied extensively since there is much interest in finding efficient methods for its resolution and several methods have been proposed to solve the problem, among them graph coloring, the simulated path, genetic algorithms, among others [5]–[7].

The aim of this work is to propose a new method for solving the problem of assigning class schedules based on Graph Coloring and Boolean Satisfaction (SAT). The proposal to attack the problem of assigning class schedules in Educational Institutions is prevent two or more teachers from being assigned several subjects or groups at the same time in a single day. The graph coloring algorithm is suitable because it proposes a feasible solution since it is possible to do the implementation by substituting the color entity for the groups of an Educational Institution. After this, a solution is proposed through Boolean Satisfaction (SAT); that is, it will show how to translate the vertices and edges of a graph (G) to a set of clauses in Conjunctive Normal Form (FNC). After having all the clauses in Conjunctive Normal Form, the DPLL algorithm (Davis-Putnam-Logemann-Loveland) will be applied to find the SAT solution, to find the truth values that make the formula in Conjunctive Normal Form satisfactory [2], [8], [9].

In this document the basic concepts about Boolean Satisfaction (SAT), graph coloring and complete algorithms for the SAT solution are presented. Some of the different methods that exist for solving the class schedule problem through SAT and graph coloring will be presented. At the end, the implementation in a software project, the results obtained, and the respective conclusions and future work are shown.

II. THEORETICAL BACKGROUND

To solve the assigning class schedules problem, various algorithms have been designed and solution strategies have been applied. This paper explains the use of techniques based on the SAT satisfiability problem and Graph Coloring; however, it is worth mentioning only a few solution strategies that have been implemented so far. This section will briefly discuss an overview of the main techniques (algorithms or sets of algorithms) and strategies (solution development).

In *Academic Assignment Model of Loads and Organization of Schedules for UJAT*, the scheduling problem was represented, modeled as an Optimization problem. So many hard and soft constraints are modeled for the problems of academic load assignment (ACA) and class schedules organization (OHC), as well as the definition of an objective function for each problem [10]. This model is show in the equations (1) to (8).

- ***Academic load assignment (ACA).***

$$\max Z = \sum Scv. \quad (1)$$

$$st$$

$$Hcv = 0 \quad (2)$$

$$Svc = weak\ restrictions. \quad (3)$$

$$Hcv = Strong\ restrictions. \quad (4)$$

- ***Class schedules organization (OCH)***

$$\min Z = Scv. \quad (5)$$

$$st$$

$$Hcv = 0 \quad (6)$$

$$Svc = weak\ restrictions. \quad (7)$$

$$Hcv = Strong\ restrictions. \quad (8)$$

In *Tabu Search-Based Algorithm for Class Scheduling Problem*, an algorithm based on the Tabu Search metaheuristic is presented to solve the class schedules assigning problem at the Computer Department of the Experimental Faculty of Sciences and Technology of the University of Carabobo Venezuela. The experimental results showed that, in general, the developed algorithm produces better solutions than the manual method, in terms of the quality of the schedules obtained and the amount of time used to generate them [11].

Academic Load Allocation Model using Genetic Algorithms presents a computational model capable of finding the optimal assignment of classes, teachers and schedules using a genetic algorithm. Tests were carried out on the model, taking as a basis the requirements in each period of the Computer Systems Engineering (ISC) career at the Nuevo Laredo Technological Institute (ITNL) and a list of available teachers and their schedule, finding better results than those obtained manually [12].

Evolutionary algorithms for the resolution of a Timetabling problem research method focuses on finding a solution to the schedules allocation problem that exists in the teaching hours of Engineering in Applied Computer Science and Civil Engineering Computer Science of the University of Valparaíso. The solution must satisfy a restriction set, which occur frequently in the academic environment. To solve this assignment problem, Evolutionary Algorithms were used, which belong to the group of meta-heuristic techniques; those techniques are methods that consist in systematizing ideas, in order to develop efficient algorithms capable of delivering optimal solutions to the problem of assigning schedules [13].

In this research: *Integer Programming Models for a Schedule Problem for Universities*, a problem of Schedule Programming in Universities has been characterized, modeled, and solved through Linear Programming, obtaining Models and Methods that allow solving large problems in reasonable Computational Times. Four Solving Methods based on Integer Linear Programming were proposed to solve the problems. Two Methods can obtain optimal solutions for any instance where a solution exists. The first Method directly assigns classes to periods and classrooms (Timetabling); the second assigns the classes to periods and Classroom Types that later, by the algorithm, assigns Classroom Types to specific Classrooms that belong to each Type (Timetabling with Classroom Types). These two named Methods allow problems solving satisfying a desired Quality Level but in a high Computational Time, existing large problems, in which it was not possible to obtain results in a reasonable Computational Time [14].

For this reason, two Methods were proposed that do not guarantee obtaining optimal solutions and even finding a solution, for instances in which there may be a solution. These Methods are based on relaxing restrictions in a first stage that increase the complexity of the problem, obtaining solutions that allow setting variables and solving smaller problems for each day in a second stage. Similarly, one Method includes assigning directly to each classroom (Timetabling with Relaxation Strategy) and another, initially assigning to Classroom Types (Timetabling with Classroom Types and Relaxation Strategy). For each proposed problem, a solution was found in a reasonable Computational Time and satisfying a desired Quality Level, for the last two Methods named above.

University of Manizales Planning Software. This software that works under Windows operating systems, was developed with the FoxPro database manager, this software allows manual, semiautomatic, automatic assignments of classrooms and class schedules, also allows the management of teachers, buildings, and faculties [15]–[17].

III. METHODOLOGY

The software for assignment of class schedules through Boolean satisfiability was fully developed in the Visual Basic 6.0 programming language, this software was chosen because it is a powerful programming environment, with which it can be built quickly and efficiently applications for Microsoft Windows. Regarding the database, it is necessary to point out that the results will be stored in an Access file and will be read by Visual Basic and will show the results in a user-friendly manner. The purpose is to solve the problem of assigning class schedules in Educational Institutions to prevent two or more teachers from being assigned several classes or groups at the same time in a single day.

The graph coloring algorithm is suitable because can propose a feasible solution substituting the color entity for the class groups of an Educational Institution. After this, a solution is

created through Boolean Satisfaction (SAT); that is, it will show how to translate the vertices and edges of a graph (G) to a set of clauses in Conjunctive Normal Form (CNF). After having all the clauses in Conjunctive Normal Form, the DPLL algorithm (Davis-Putnam-Logemann-Loveland) will be applied to find the SAT solution, to satisfy the truth values that make TRUE the formula in Conjunctive Normal Form.

Guidelines for the AHOSAT Solution. a) Conversion of the problem from Assigning class schedules to Graphs. The initial conditions are: The schedule for three classes is generated, the three classes are taught by three different teachers, the number of groups that receive the classes are three, the number of days in the schedule is one and the number of hours that the groups receive per day are also three. For the representation in the software, we use one of all possible combinations (In fig. 1 we show the problem as a graph).

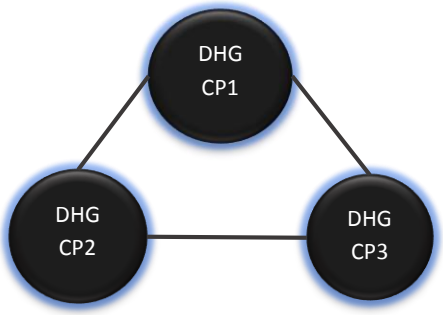

Fig. 1 Graph initial conditions. Source: Author

Reduction of the graph to the Conjunctive Normal Form (FNC): The graph must be converted to clauses and SAT instances and this reduces a graph to clauses in conjunctive normal form (FNC). In this state, the full DPLL method is applied to solve the SAT satisfiability problem. Once the truth values that satisfy the formula have been assigned, the variables values are interpreted in the context of graphs from which the color number is obtained, as well as the graph coloring.

The equation (9) is the general clause to solve the problem. D = day, H = hour, G = Group, M = Class and P = Teacher

$$DKHiGiMiPi \text{ where } k = 1 \text{ and } i = 1,2,3 \quad (9)$$

The restriction is that any class can be assigned on any day at any time and the clauses are grouped as shown in equation (10).

$$\begin{aligned}&(D1H1G1M1P1 \text{ v } D1H1G1M2P2 \text{ v } D1H1G1M3P3) \text{ ^}\\&(D1H2G1M1P1 \text{ v } D1H2G1M2P2 \text{ v } D1H2G1M3P3) \text{ ^}\\&(D1H3G1M1P1 \text{ v } D1H3G1M2P2 \text{ v } D1H3G1M3P3) \text{ ^}\\&(D1H1G2M1P1 \text{ v } D1H1G2M2P2 \text{ v } D1H1G2M3P3) \text{ ^}\\&(D1H2G2M1P1 \text{ v } D1H2G2M2P2 \text{ v } D1H2G2M3P3) \text{ ^}\\&(D1H3G2M1P1 \text{ v } D1H3G2M2P2 \text{ v } D1H3G2M3P3) \text{ ^}\\&(D1H1G3M1P1 \text{ v } D1H1G3M2P2 \text{ v } D1H1G3M3P3) \text{ ^}\\&(D1H2G3M1P1 \text{ v } D1H2G3M2P2 \text{ v } D1H2G3M3P3) \text{ ^}\\&(D1H3G3M1P1 \text{ v } D1H3G3M2P2 \text{ v } D1H3G3M3P3)\end{aligned} \quad (11)$$

A teacher can only give a class to a group on the same day at a certain time (equation (12))

$$\begin{aligned}&(\sim D1H1G1M1P1 \text{ v } \sim D1H1G2M1P1) \text{ ^}\\&(\sim D1H1G1M1P1 \text{ v } \sim D1H1G3M1P1) \text{ ^}\\&(\sim D1H1G2M1P1 \text{ v } \sim D1H1G3M1P1) \text{ ^}\\&(\sim D1H2G1M1P1 \text{ v } \sim D1H2G2M1P1) \text{ ^}\\&(\sim D1H2G1M1P1 \text{ v } \sim D1H2G3M1P1) \text{ ^}\\&(\sim D1H2G2M1P1 \text{ v } \sim D1H2G3M1P1) \text{ ^}\\&(\sim D1H3G1M1P1 \text{ v } \sim D1H3G2M1P1) \text{ ^}\\&(\sim D1H3G1M1P1 \text{ v } \sim D1H3G3M1P1) \text{ ^}\\&(\sim D1H3G2M1P1 \text{ v } \sim D1H3G3M1P1) \text{ ^}\\&(\sim D1H1G1M2P2 \text{ v } \sim D1H1G2M2P2) \text{ ^}\\&(\sim D1H1G1M2P2 \text{ v } \sim D1H1G3M2P2) \text{ ^}\\&(\sim D1H1G2M2P2 \text{ v } \sim D1H1G3M2P2) \text{ ^}\\&(\sim D1H2G1M2P2 \text{ v } \sim D1H2G2M2P2) \text{ ^}\\&(\sim D1H2G1M2P2 \text{ v } \sim D1H2G3M2P2) \text{ ^}\\&(\sim D1H2G2M2P2 \text{ v } \sim D1H2G3M2P2) \text{ ^}\\&(\sim D1H3G1M2P2 \text{ v } \sim D1H3G2M2P2) \text{ ^}\\&(\sim D1H3G1M3P3 \text{ v } \sim D1H3G3M3P3) \text{ ^}\\&(\sim D1H3G2M3P3 \text{ v } \sim D1H3G3M3P3) \text{ ^}\\&(\sim D1H1G1M3P3 \text{ v } \sim D1H1G2M3P3) \text{ ^}\\&(\sim D1H1G1M3P3 \text{ v } \sim D1H1G3M3P3) \text{ ^}\\&(\sim D1H1G2M3P3 \text{ v } \sim D1H1G3M3P3) \text{ ^}\\&(\sim D1H2G1M3P3 \text{ v } \sim D1H2G2M3P3) \text{ ^}\\&(\sim D1H2G1M3P3 \text{ v } \sim D1H2G3M3P3) \text{ ^}\\&(\sim D1H2G2M3P3 \text{ v } \sim D1H2G3M3P3) \text{ ^}\\&(\sim D1H3G1M3P3 \text{ v } \sim D1H3G2M3P3) \text{ ^}\\&(\sim D1H3G1M3P3 \text{ v } \sim D1H3G3M3P3) \text{ ^}\\&(\sim D1H3G2M3P3 \text{ v } \sim D1H3G3M3P3)\end{aligned} \quad (12)$$

Only one class can be taught in the combination of day, time, and group (equation (12)).

$$\begin{aligned}&(\sim D1H1G1M1P1 \text{ v } \sim D1H1G1M2P2 \text{ v } \sim D1H1G1M3P3) \text{ ^}\\&(\sim D1H1G1M1P1 \text{ v } \sim D1H1G1M2P2 \text{ v } \sim D1H1G1M3P3) \text{ ^}\\&(\sim D1H3G1M1P1 \text{ v } \sim D1H3G1M2P2 \text{ v } \sim D1H3G1M3P3) \text{ ^}\\&(\sim D1H1G2M1P1 \text{ v } \sim D1H1G2M2P2 \text{ v } \sim D1H1G2M3P3) \text{ ^}\\&(\sim D1H2G2M1P1 \text{ v } \sim D1H2G2M2P2 \text{ v } \sim D1H2G2M3P3) \text{ ^}\\&(\sim D1H3G2M1P1 \text{ v } \sim D1H3G2M2P2 \text{ v } \sim D1H3G2M3P3) \text{ ^}\\&(\sim D1H1G3M1P1 \text{ v } \sim D1H1G3M2P2 \text{ v } \sim D1H1G3M3P3) \text{ ^}\\&(\sim D1H2G3M1P1 \text{ v } \sim D1H2G3M2P2 \text{ v } \sim D1H2G3M3P3) \text{ ^}\\&(\sim D1H3G3M1P1 \text{ v } \sim D1H3G3M2P2 \text{ v } \sim D1H3G3M3P3)\end{aligned} \quad (12)$$

A class can only be taught for one hour on the same day (equation (13)).

(∼D1H1G1M1P1 v ∼D1H2G1M1P1) ^
(∼D1H1G1M1P1 v ∼D1H3G1M1P1) ^
(∼D1H2G1M1P1 v ∼D1H3G1M1P1) ^
(∼D1H1G2M1P1 v ∼D1H2G2M1P1) ^
(∼D1H1G2M1P1 v ∼D1H3G2M1P1) ^
(∼D1H2G2M1P1 v ∼D1H3G2M1P1) ^
(∼D1H1G3M1P1 v ∼D1H3G3M1P1) ^
(∼D1H1G3M1P1 v ∼D1H3G3M1P1) ^
(∼D1H2G3M1P1 v ∼D1H3G3M1P1) ^
(∼D1H1G1M2P2 v ∼D1H2G1M2P2) ^
(∼D1H1G1M2P2 v ∼D1H3G1M2P2) ^
(∼D1H2G1M2P2 v ∼D1H3G1M2P2) ^
(∼D1H1G2M2P2 v ∼D1H2G2M2P2) ^
(∼D1H1G2M2P2 v ∼D1H3G2M2P2) ^ (13)
(∼D1H2G2M2P2 v ∼D1H3G2M2P2) ^
(∼D1H1G3M2P2 v ∼D1H3G3M2P2) ^
(∼D1H1G3M2P2 v ∼D1H3G3M2P2) ^
(∼D1H2G3M2P2 v ∼D1H3G3M2P2) ^
(∼D1H1G1M3P3 v ∼D1H2G1M3P3) ^
(∼D1H1G1M3P3 v ∼D1H3G1M3P3) ^
(∼D1H2G1M3P3 v ∼D1H3G1M3P3) ^
(∼D1H1G2M3P3 v ∼D1H2G2M3P3) ^
(∼D1H1G2M3P3 v ∼D1H3G2M3P3) ^
(∼D1H2G2M3P3 v ∼D1H3G2M3P3) ^
(∼D1H1G3M3P3 v ∼D1H3G3M3P3) ^
(∼D1H1G3M3P3 v ∼D1H3G3M3P3) ^
(∼D1H2G3M3P3 v ∼D1H3G3M3P3)

Once the problem is presented in clauses in Conjunctive Normal Form, the DPLL algorithm is applied for the SAT solution, to find the truth values that satisfy the formula. The pseudocode is shown at algorithm 1.

Algorithm 1**. Pseudocode select and simplify**

```
Select (var, V): var is selected from a set V of
proportional variables, once selected, var is removed
from the set V.
Var = applied clause
Value = 1 or 0
S = complete set
Ss = Return value (S simplified)
C = Each clause.
Satisfactible(S):
If S = = {1} then true,
Else false
End
Insatisfactible(S):
If S = = {0} then true,
Else false
End
Simplify(var, valor, S, Ss):
Ss = ∅;
If valor = = 1 then
        If S = = {C} ∧ var ∈ C then Ss = {1}
        // if the set has only one clause, return set 1
        otherwise retrieve each clause.
        Else for all C ∈ S {
                If var ∈ C then;
                // if it is contained with the same sign,
                the entire group is eliminated.
                Else if ¬var C then
                // if it is contained but with the
                opposite sign, it removes only the
                clause.
                {
                Cs = C – {¬var};
                If Cs = = ∅ then {
                        Ss = {0};
                        Break;}
                Else Ss = Ss ∪{Cs}
                }
        Else Ss = Ss ∪{C}}
Else if valor = = 0 then
        If S = = {C} ∧ ¬var ∈ C then Ss = = {1}
        Else for all C ∈ S
        If var ∈ C then {
                If C = {var} then {
                        Ss = {0};
                        Break;}
                Else {
                Cs = C – {var};
        Ss = Ss ∪{Cs}}
Else if ¬var ∈ C then;
        Else Ss = Ss ∪{C}
```

Inputs: S: Set of Clauses. It is assumed that in S there are no repeating clauses, and that no clause has repeating literals, V: Set of propositional variables that occur in S. **Outputs**: "Satisfactory", "Unsatisfactory". See algorithm 2.

Algorithm 2**. Pseudocode DPLL**

```
DPLL(S):
 If S = {{1}} then return "Satisfactible",
If {0} ∈ S then return "Insatisfactible",
DPLL2: if {L} ∈ S then {
        If L = var then simplify (var, 1, S, Ss)
        Else if L = ¬var then simplify (var, 0, S, Ss)
DPLL(Ss)}
DPLL3:{ Select (var, V);
Simplify (var, 1, S, Ss)
If DPLL(Ss)="Satisfactible" then return ("satisfactible");
Else if DPLL (Ss) = "Insatisfactible" then
        {Simplify (var, 0, S, Ss)
        DPLL(Ss)}}
```

IV. IMPLEMENTATION

To solve the schedule generating problem, data type structures and function modules were created in Visual Basic programming language necessary to implement the Davis-Putman SAT solution algorithm were used.

The list of data structures and function modules created for the project are a list of structures: Clause and condition structure and a list of function modules: GenerationFNC and AlgorithmDP

Structures are complex data types in a programming language that are used to represent sets of data that can be manipulated by the program as complex units made up of different data types.

The Clause structure is used to represent a variable within the universe of schedule variables, this variable is composed of the

following definitions: a unique "identifier" value used to record the clause, a "day" value that represents the day on which the variable can be assigned within the schedule, a value "hour" that represents the time of day in which the variable can be assigned within the schedule, a value of "user" that represents the teacher's code, a value "class" That represents the class code and a "group" value that represents the group code to which the schedule will be assigned. The structure in code of the Visual Basic programming language as shown in algorithm 3.

Algorithm 3**. Public type clause**

```
Public Type clausula
        identifier As Long
        day As Long
        hour As Long
        user As Long
        class As Long
        group As Long
End Type
```

The condition structure is used as a data type that represents a condition that must be fulfilled for the hour assignment through the SAT solution. The structure must contain an identifier that represents the clause unique number that will be identified with the condition, a sign value (1 or -1, true or false respectively) that represents the Boolean result after evaluating the condition, a value that represents the group to which the condition belongs and a constraint value that represents the constraint number to which it belongs. This structure is used directly by manipulating the Boolean value to find the solution to the time assignment problem with the Davis-Putman SAT algorithm. The structure in code of the Visual Basic programming language as shown in algorithm 4.

Algorithm 3**. Public type condition**

```
Public Type condition
        identifier As Long
        sign As Long
        restriction As Long
        group As Long
End Type
```

The GenerationFNC.bas module contains the necessary functions to generate the group of conditions in conjunctive normal form FNC that will be evaluated by the Davis-Putman SAT solution algorithm. The module contains the function Generation_FNC that generates the conditions structure in FNC (conjunctive normal form). For the Davis-Putman SAT algorithm used to solve the problem of generating schedules, the different conditions must appear in FNC, below, a pseudo-code is shown to illustrate how the structure generated by the function is at equation (14):

```
{X11 v X12}^{X21 v X22}^{X31 v X32}^{X41 v X42}^
 {~X11 v ~X12} ^ {~X21 v ~X22} ^ {~X31 v ~X32} ^
 {~X41 v ~X42} ^ {~X11 v ~X31} ^ {~X12 v ~X32} ^ (14)
 {~X11 v ~X21} ^ {~X12 v ~X22} ^{~X21 v ~X41} ^
 {~X22 v ~X42} ^ {~X31 v ~X41} ^ {~X32 v ~X42}
```

Where *X11, X12, X21 ... X42* are problem conditions that will be evaluated by the Davis-Putman SAT algorithm, the symbol "^" represents the logical conjunction of variables (in this case of conditions) and "v" represents the logical disjunction variable (in this case of conditions). The prototype of the Generation_FNC function is Public Sub Generation_FNC()

The getClause function is responsible for returning a clause stored in an array of clauses or variables set for the problem, considering the identifier. The values entered by parameters are: "value" is the unique identifier of the clause and "clauses ()" is the array that contains all the clauses or variables proposed for the problem. The prototype of the getCover function in the is Private Function getClosure (ByVal value As Integer, ByRef clauses () As clause) As clause.

The assignGroupClass function allows assigning to a condition the group to which it belongs (grouping conditions means that they are part of the FNC structure together with other conditions by means of the symbol "^" conjunction) considering premises such as the day, the group, and the class to assign within the schedule. The values entered by parameters are: "conditions ()" array that contains the conditions in FNC form, "clauses()" is the array that contains all the clauses or variables set for the problem and "clause" is the variable to which group assignment is in progress. The prototype of the AssignGroup function is: Private Function assignGroupClass(ByRef condition() As condition, ByRef clausule() As clause, ByRef clauseActual As clause) As Integer

The function assignGroupProfesor allows assigning to a condition the group to which it belongs, considering premises such as the day, the time, and the teacher to be assigned within the schedule. The values entered by parameters are: "conditions ()" array that contains the conditions in FNC form, "clauses ()" is the array that contains all the clauses or variables set for the problem and "clause" is the variable to which group assignment is in progress. The prototype for the assignTeacherGroup function is: Private Function assignGroupProfesor (ByRef conditions() As condition, ByRef clausule() As clause, ByRef clauseActual As clause) As Integer.

The function assignGeneralGroup allows assigning to a condition the group to which it belongs, considering premises such as day, time, and class. The values entered by parameters are: "conditions ()" array that contains the conditions in FNC form, "clauses ()" is the array that contains all the clauses or variables proposed for the problem and "clause" is the variable to which group assignment is in progress. The prototype of the assignGeneralGroup function is: Private Function assignGeneralGroup (ByRef conditions () As condition, ByRef clauses () As clause, ByRef clauseActual As clause) As Integer.

The fitArrayString function is a utility function that allows to resize the String array to fit the desired size considering the number of String elements. The values entered by parameters are: "strings ()" which is the array of String to be adjusted and "quantity" which is the size to which the arrangement is to be adjusted. The prototype of the fitArrayString function is: Public Function fitArrayString (Strings () As String, Amount as Integer) As String ()

The function fitArrayConditions is a utility function that allows you to resize the Condition array to fit the desired size

considering the number of elements that the Condition structure contains. The values entered by parameters are: "conditions ()" which is the arrangement that contains the elements of the condition structure to be adjusted and "quantity" which is the size to which the arrangement is to be adjusted. The prototype of the function adjustArrayConditions is: Public Function adjustArrayConditions (conditions () As condition, quantity As Integer) As condition ()

The function adjustArrgloClausulas is a utility function that allows you to change the size of an arrangement type Clauses to fit the desired size considering the number of elements of the Clause structure it contains. The values entered by parameters are: "clauses ()" which is the arrangement that contains the elements of the clause structure to be adjusted and "quantity" which is the size to which the arrangement is to be adjusted. The prototype of the function fitArrangeClausulas is: Public Function fitArrangeClausulas (clauses () As clause, quantity As Integer) As clause ()

The findIndex function allows you to find the location of a subject within the group of subjects established for the group. The main purpose of this function is to help generate the clauses or variable that will be resolved by the Davis-Putman SAT Algorithm, where each variable is represented by the values day, hour, teacher, class, and group (*DiHiGiXiMi)* where *i = 0 ... n* for each variable. The prototype of the findIndex function is: Private Function findIndex (userxassignment () As String, index As Integer, key As String) As Integer

DPL algorithm module contains a set of functions that will be detailed. The initiators function allows to initialize the arrays of clauses and conditions that will be used to store the variables and their results after evaluating the different variables (translated into conditions) with the Davis-Putman SAT algorithm. The prototype of the initiators function is: Public Sub initiators ()

The AlgorithmDPLL function represents the Davis-Putman SAT algorithm that evaluates the different conditions (they represent clauses or variables) and generates a satisfiability result that allows generating a Class Schedule. The values entered for the parameters are: "conditions ()" array containing the conditions in FNC form. The prototype of the AlgorithmDPLL function is: Public Function AlgorithmDPLL (ByRef conditions () As condition) As String.

The function writeSolution represents the first utility function of the Davis-Putma SAT algorithm, its main task assigning the Boolean value (1 or -1 for true and false respectively) assigned by the evaluation of the Davis-Putman SAT algorithm for a specific condition. The values entered for the parameters are: "condition" represents the condition to which the Boolean value will be assigned, and "value" represents the Boolean value to be assigned: Public Sub writeSolution (conditionActual As condition, ByVal value As Integer)

The Simplify function represents the second utility function for the Davis-Putman SAT algorithm, its main task is to simplify a specific group of conditions considering a Boolean value (true or false) for any of the conditions that is part of the group. The values entered for the parameters are: "conditionEvaluated" that represents the condition for which the Boolean value will be assigned, "value" Boolean (1 or -1 for true and false respectively) and "conditions ()" array containing the conditions in FNC form. The prototype of the quantityGroup function is: Public Function Simplify (conditionEvaluated As condition, ByVal value As Integer, conditions () As condition) As condition (). This function receives the condition to be evaluated, the value that is assigned to the condition and the group of conditions. This returns the simplified group of conditions.

The group quantity function is a utility function and indicates whether the first condition belongs to a group with a single element, within the set of grouped conditions. The values for the entered parameters are: "conditions ()" array containing the conditions in FNC form. The prototype of the quantityGroup function is: Private Function quantityGroup (conditions () As condition) As Integer

The getSoloGroup function is a utility function that allows you to obtain the identifier of the group to which the current condition that is being evaluated with the Davis-Putman SAT algorithm belongs. The values entered for the parameters are: "identifier" unique value that identifies the condition (same identifier as the clause) and "idGroup" value that represents the unique identifier of the group and "conditions ()" array that contains the condition type elements that they are evaluated by the SAT algorithm. The prototype of the getSoloGroup function is: Private Function getSoloGroup (ByVal identifier As Integer, ByVal groupid As Integer, conditions () As condition) As condition ()

The validateGroup function is useful and allows you to validate if a group (grouping of conditions) has already been previously omitted by the evaluation of the Davis-Putman SAT algorithm. The values entered by parameters are: "group" unique identifier of the group, "groupOmitidos ()" array that contains the identifiers of the groups already omitted by the SAT algorithm and "quantity" number of elements in the array of omitted groups. The prototype of the validateGroup function is: Private Function validateGroup (ByVal group As Integer, ByRef groupsOmitidos () As Integer, ByVal amount As Integer) As Integer

VI. EXPERIMENTAL RESULTS

The SAT Boolean satisfiability-based class scheduling algorithm was implemented in the Microsoft Visual Basic 6.0 programming language and Microsoft Access. The program was tested with several test cases to determine the solutions quality and in turn, compare the results with those obtained through the manual method.

The following test was carried out for six groups of Marco Fidel Suarez Educational Institution of the Municipality of Andes, on any given day of the week with six teachers who teach each of the classes. The software presented a feasible schedule to be taught at the institution. The software solution can be seen at table 1.

Table 1. Assignment of class schedules for the 10th and 11th groups, with their respective teachers on one day of the week. Source: Author

	Group					
H	**11A**	**11B**	**11C**	**10A**	**10B**	**10C**
1	Gym - T1	English - T2	Computers - T5	Math - T6	Chemistry - T4	Spanish - T3
2	Spanish - T3	Computers - T5	English - T2	Chemistry - T4	Math - T6	Gym - T1
3	Computers - T5	Math - T6	Chemistry - T4	Spanish - T3	Gym - T1	English - T2
4	English - T2	Chemistry - T4	Math - T6	Gym - T1	Spanish - T3.	Computers - T5
5	Chemistry - T4	Spanish - T3	Gym - T1	English - T2	Computers - T5	Math - T6
6	Math - T6	Gym - T1	Spanish - T3	Computers - T5	English - T2	Chemistry - T4

VII. CONCLUSIONS

The problem of assigning class schedules is extremely complicated due to the restrictions that vary according to the policies and rules of the educational institution where it is required, as well as the criteria with which said restrictions are applied.

The generated model allows obtaining schedules for each teacher and each group of students. Although the problem was modeled and solved specifically for the educational institution Marco Fidel Suárez of the Municipality of Andes Antioquia; The model allows adjusting to many scheduling problems in different educational institutions, providing great flexibility in resolution. The algorithm based on graph coloring and Boolean satisfiability SAT for assigning class schedules has been developed, tested with real data, and compared with results obtained through the manual method.

The experiments carried out show that, in all the test cases, the algorithm based on graph coloring and Boolean satisfiability SAT finds better solutions than those obtained with the manual method, in a reasonable computational time. This leads us to think that the algorithm should, in general, find good solutions to the problem of assigning class schedules at the educational institution in the municipality of Andes. Comparisons in the distribution of academic load obtained by the satisfaction algorithm and the manual process, showed that the solution proposed by the SAT algorithm is better.

For future work, the following recommendations are given research for other possible secondary restrictions, and their corresponding terms in the cost model, to improve the quality of schedules.

REFERENCES

[1] G. Aloupis, E. Demaine, and a Guo, "Classic nintendo games are (np-) hard," arXiv Prepr. arXiv1203.1895, pp. 1–21, 2012, [Online]. Available: http://arxiv.org/abs/1203.1895.

[2] A. I. Diveev, O. V. Bobr, D. E. Kazaryan, and O. Hussein, "Some methods of solving the NP-difficult problem of optimal schedule for the university," in Procedia Computer Science, 2019, vol. 150, pp. 410–415, doi: 10.1016/j.procs.2019.02.071.

[3] P. Kalla, Z. Zeng, and M. J. Ciesielski, "Strategies for solving the Boolean satisfability problem using binary decision diagrams," J. Syst. Archit., vol. 47, pp. 491–503, 2001.

[4] W. Zhang and R. E. Korf, "A study of complexity transitions on the asymmetric traveling salesman problem," Artif. Intell., vol. 81, no. 1–2, pp. 223–239, 1996, doi: 10.1016/0004-3702(95)00054-2.

[5] T. Januario, S. Urrutia, C. C. Ribeiro, and D. De Werra, "Edge coloring: A natural model for sports scheduling," Eur. J. Oper. Res., vol. 254, no. 1, pp. 1–8, 2016, doi: 10.1016/j.ejor.2016.03.038.

[6] Z. Zaeniah and S. Salman, "Designing class schedule information system by using taboo-search method," J. Pilar Nusa Mandiri, vol. 16, no. 2, pp. 241–248, 2020, [Online]. Available: https://doi.org/10.33480/pilar.v16i2.1661.

[7] V. F. Suárez, Á. Guerrero, and O. D. Castrillón, "Programación de horarios escolares basados en ritmos cognitivos usando un algoritmo genético de clasificación no-dominada, NSGA-II," Inf. Tecnol., vol. 24, no. 1, pp. 103–114, 2013, doi: 10.4067/S0718-07642013000100012.

[8] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT modulo theories: From an abstract davis - putnam - logemann - loveland procedure to DPLL(T)," J. ACM, vol. 53, no. 6, pp. 937–977, 2006, doi: 10.1145/1217856.1217859.

[9] H. Youness, M. Osama, A. Hussein, M. Moness, and A. M. Hassan, "An Effective SAT Solver Utilizing ACO Based on Heterogenous Systems," IEEE Access, vol. 8, pp. 102920–102934, 2020, doi: 10.1109/ACCESS.2020.2999382.

[10] J. L. Gomez and J. F. Solis, "Modelos de asignación de cargas académicas y organización de horarios para la UJAT," Perspect. docentes, no. 34, pp. 39–48, 2007.

[11] A. A. Martínez, "Algoritmo basado en tabu search para el problema de asignación de horarios de clases," Univ. Carabobo, p. 11, 2012.

[12] J. B. López Bruno, "Modelo de asignación de carga académica usando algoritmos genéticos," Tecnol. Nac. Mex., no. 5, pp. 1–7, 2007.

[13] J. E. Molina-Araya, "Algoritmos Evolutivos para la resolución de un problema de tipo Timetabling," Universidad de Valparaíso, 2007.

[14] A. Saldaña Crovo, C. Oliva San Martin, and L. Pradenas Rojas, "Models of Integer Programming for an University Timetabling Problem," Ingeniare., vol. 15, no. 3, pp. 245–259, 2007.

[15] R. Baker, B. Evans, Q. Li, and B. Cung, "Does Inducing Students to Schedule Lecture Watching in Online Classes Improve Their Academic Performance? An Experimental Analysis of a Time Management Intervention," Res. High. Educ., vol. 60, no. 4, pp. 521–552, 2019, doi: 10.1007/s11162-018-9521-3.

[16] J. C. Marín Ángel and P. A. Maya Duque, "Modelo lineal para la programación de clases en una institución educativa," Ing. y Cienc., vol. 12, no. 23, pp. 47–71, 2016, doi: 10.17230/ingciencia.12.23.3.

[17] R. Hernández, J. Miranda P, and P. A. Rey, "Programación de Horarios de Clases y Asignación de Salas para la Facultad de Ingeniería de la Universidad Diego Portales Mediante un Enfoque de Programación Entera," RIS - Rev. Ing. Sist., vol. 22, no. 1, pp. 121–141, 2008, [Online]. Available: http://www.dii.cl/~ris/tabla.php.