SOLVING SCHEDULING PROBLEM WITH GENETIC ALGORITHM APPROACH

Andriamirindra Rakotomahefa [1], Falimanana Randimbindrainibe [2], Matio Robinson [3]

[1] *PhD student, SCA, ED-STII / University of Antananarivo, Antananarivo, Madagascar*
[2] *Professor, SCA, ED-STII, Antananarivo, Madagascar*
[3] *MC, SCA, ED-STII, Antananarivo, Madagascar*

ABSTRACT

This paper contributes to modeling and solving the scheduling problem that belongs to the NP-complete complexity class. The data are manipulated and structured with UML principles and they are processed according to the genetic algorithm approaches. Markov Chain Modeling is a way of demonstrating and ensuring the convergence of the algorithm. The study and application of schema theories on crossover and mutation operators improve both the final solution quality and the execution time duration. The results make it possible to evaluate the temporal complexity which is of the form ax + b. This proves and confirms that the objective of this search is achieved: the scheduling problem (NP-complete) is reduced to a polynomial problem O (n).

Keyword : *Scheduling, Genetic Algorithm, Timetable Constraints, Algorithm Complexity, Optimization, Schema Theories.*

---

1. INTRODUCTION

A timetable is defined as a table of information showing when certain events are scheduled to take place. A timetable has always been useful in every Organization that its work in a production process would be well organized. Timetabling arises in a wide variety of domains. Such as transportation problem, machine scheduling problem, sports events, staff scheduling such as the nurse scheduling problem and educational timetabling. Problems related to constructing a timetable as stated in [1] is NP-complete, which means it is a very tough problem that one would stand out in.

Wren [2] defines the timetabling problem as a special case of scheduling: Timetabling is the allocation, subject to constraints, of given resources to be placed in space time, in such a way as to satisfy as possible a set of desirable objectives. According to Burke et al [3] and Burke and Petrovic [4], a number of constraints must be satisfied to build a timetable. These constraints are usually classified into two types:

- Hard constraints are those that must be satisfied in order for the timetabling problem to be feasible. Hard constraints have to be satisfied under any circumstances,
- Soft constraints are those whose violation should be minimized in order to produce the best timetable. Soft constraints need to be satisfied as much as possible.

Economic efficiency, costs and resource utilization are also important drivers for improved timetable generation.

The timetabling problem for educational organizations constitutes a sub-category of scheduling problems. It can be classified into three types, each with their own specific characteristics and constraints:

- School timetabling: These are problems that are concerned with assigning the weekly lessons in schools. The aim is to allocate a set of teachers to a set of classes (groups of students) for a set of lessons in a set of periods, while satisfying a set of constraints.
- Exam timetabling: The main objective is to assign a set of exams to a given set of time slots. Each exam has a list of enrolled students. The key constraint is that no student can possibly sit in more than one exam at the same time.
- Course timetabling: The purpose is to assign courses and associated events, groups of students and lecturers to time slots in such a way that no conflict occurs at any period. The number of assigned students should not exceed the maximum recommended number in a room.

Many scientists perform studies on educational timetable problem due to its complexities. In this research, our focus is to establish the algorithm to automate and to boost a scholastic timetable.

2. PROBLEM FORMULATION

2.1 Problem description

The school timetable is affected by many parameters and must satisfy a large number of constraints. The entities that are involved in such a timetable are the teachers, the courses, the classes and the time slots. Therefore, constructing a timetable means to assign, for the entire teacher-course-class triples, the time slots that the teaching process will take place.

All the necessary input data must be available before constructing the timetable. This data will provide all the necessary information about the relations between teachers, classes and courses. This information comprises: the total number of classes, the total number of teachers, the number of weekly teaching days, the maximum number of daily hours, the total number of courses per class, the courses taught by each teacher as well as the classes to which these courses are taught, the maximum number of weekly hours teaching, the days each teacher is available at school, the maximum number of authorized consecutive times slot, the total number of classrooms.

Hard constraints that must be satisfied in order to keep the timetable valid are the following:

- H1: All subjects must be included in the schedule.
- H2: Each teacher must not be assigned to more than one class in a specific time slot.
- H3: Each class must not have more than one course in a specific time slot.
- H4: Each classroom should not take two or more classes in a specific time slot.
- H5: Each subject should be planned according to the teacher availability
- H6: Each course must be offered in a suitable room
- H7: A course belonging to a common core must be planned for the same period for all classes concerned.
- H8: For any course that is assigned more than one period in a day (block), these periods must be consecutive.

Soft constraints that should be satisfied in order the timetable to be considered of high quality are the following:

- S1: Each class should not have a single course in a day.
- S2: Classes should not have long idle hours between two courses
- S3: Each classes should not exceed the maximum number of daily hours
- S4: Each teacher should not exceed the maximum number of daily hours.
- S5: The classroom assigned to a subject must be the same throughout the week.
- S6: The number of students must be less than or equal to the capacity of the classroom where the course is held.

Scheduling means assigning for the triplets <*teacher - course - class*> the time's periods and the place where the teaching will take place.

We define

$$f: \mathcal{P}, \mathcal{M}, \mathcal{C} \longrightarrow \mathcal{T}$$
$$P, M, C \longmapsto f(P, M, C) = T$$

such as

- P : set of teachers
- M : set of courses
- C : set of classes
- T : set of periods

Then, the problem consists to:

$$\text{Optimize } f(P, M, C) = T_i \quad \text{where } i \in [1, nt] \tag{1}$$

2.2 Related works

Since the 1960s, much research has been devoted for scheduling problem and direct heuristic was the first approach. Many techniques have been proposed to solve the timetabling problem such as graph coloring problem, integer linear programming technique, logic programming, decision-support systems, specific heuristics such as simulated annealing, tabu search, ant colony algorithms, genetic algorithms, multi-neighborhood local search, multiagent methods, particle swarm optimization, great deluge algorithm, case based reasoning. The recently emerging search techniques of hyper-heuristics have been applied to the timetabling problem with very promising results.

Overall, two types of meta-heuristics algorithms are taking place: local area search-based and population-based [5]. Each type has its own unique advantages and weaknesses. Local area-based algorithms are simulated annealing, a very large neighborhood search, tabu search, and many more. Usually, local area-based algorithms focus on the exploitation rather than exploration, which means that they move in one direction without considering any extensive scan of the search area. Population-base algorithms in the other hand would start with an amount of solutions and refining them to obtain some global optimum solutions in the exploration space. Population based algorithms that are commonly used to deal with the timetabling problems are evolutionary algorithms, particle swarm optimization, colony ant-optimization, artificial immune system.

However, genetic algorithm stands out due to its performance achievement with an ability to escape local optima and diversity [6].

Population-based algorithms, particularly genetic algorithms have been the most common solution in recent years. Several researchers amplified the efficiency of genetic algorithm using modified genetic operators and techniques LS [7]. Usually, when a simple genetic algorithm is applied, it may produce illegal timetables that have duplication and / or missing events. Quality of solutions produced by population-based algorithms may not be better than the local area-based algorithms mainly due to the fact that population-based algorithms are more concerned with exploration than exploitation.

Other approaches use hybrid heuristics that combine two or more of the above techniques [8] or offer designs built around constraint programming.

2.3 Otherview of Genetic algorithm

A Genetic Algorithm (GA) is a programming technique that mimics biological evolution as a problem solving strategy. It is a specialization of the evolutionary programs based on the principles of the normal selection and the random transformation from Darwnistic biological evolution. GA was formalized in 1975 by John Holland at the University of Michigan and have been growing in popularity since, particularly for solving problems with a large irregular search space of possible solutions [9].

Genetic Algorithm works on two types of spaces like Coding space and the Solution space. In other words, Coding space defines the Genotype space and Solution space outlines the Phenotype space. Evolutionary algorithm is the part of the evolutionary computation. Evolutionary approaches are simply defining the idea:

a) Generate a population of possible solution to the problem
b) Choose the best individuals from the population (using the methods inspired by the Survival of the Fittest)
c) Produce a new generation by combining these individuals' best one
d) Stop when the best individual of the population is found

All the GAs works on a set of populations. Each individual in population is called the chromosome. The individual characters of the chromosome are known as the Genes. The population size is very important in the GA for getting the amount of the information that is stored by the GA. The mechanisms used to perform the search and to allow the evolution to new generations are called operators. The main operators are selection, crossover and mutation.

All the chromosomes are evaluated using some criteria that help us to rank the individuals according to their relative fitness. Fitness function is also called the evaluation function.

The selection operator is then applied to choose the better individuals. The selection mechanism of genetic algorithms operates on the phenotypic level of the individuals in the population. The selection of individuals is biased towards good individuals. In the classical fitness-based roulette-wheel, the chance of an individual to be selected is based on its relative fitness in the population.

The crossover operator is responsible for the recombination process. Crossover is the main operator used for the reproduction of the population. This is used to combine two parents to create two new individuals that are called "the offspring". The offspring then would contain genetic information from both parents. According to Goldberg [10] the combination of crossover and selection operators is the core of the innovation process that explains the success of GAs.

The mutation operator performs changes in a single individual. Mutation randomly searches in the neighborhood of a particular solution. Its role is very significant to guarantee that the whole search space is reachable.

The new individuals are evaluated and they will replace the worst individuals of the current population. This process would then be recurring until a satisfactory solution is achieved or a prefixed number of generations is highlighted. The process of going from the current population to the next population forms one generation in the execution of a GA.

There are four fundamental differences between GAs and more traditional optimization and search procedures. These are in short:

- GAs work with a coding of the parameter set, not the parameters themselves.
- GAs search from a population of points, not from a single point.
- GAs use pay-off (objective function) information, not derivatives or other auxiliary knowledge.
- GAs use probabilistic transition rules, not deterministic rules.

3. PROBLEM SOLUTION

3.1 Data List

a) Class data: $\mathcal{E}c$

- D : set of departments
- N : set of levels
- n_d : total number of departments
- n_n : total number of levels

b) Time data: $\mathcal{E}t$

- J : set of weekly teaching days
- H : set of daily time slots
- T : set of weekly time slots
- n_j : total number of weekly teaching days
- n_h : total number of daily time slots
- n_t : total number of weekly time slots

c) Course data: $\mathcal{E}m$

- M : set of subjects
- V : set of weekly volumes hours of subjects
- n_m : total number of subjects

d) Teacher data: $\mathcal{E}e$

- P : set of teachers
- F : Teacher availability
- n_p : total number of teachers

e) Student data: $\mathcal{E}s$

- S : set of students
- n_s : total number of students

f) Classroom data: $\mathcal{E}r$

- R : set of classrooms
- Y : set of room types
- C_R : room capacity
- n_r : total number of classrooms
- n_y : total number of classroom types

- M_Y : set of pairs <course – type> of room

3.2 Mathematical Model

The generation X_{n+1} is a function of the current generation X_n. Among the n individuals of X_n, we create a mating pool of *k* chromosomes to be parents. These *k* chromosomes will be selected and replicated together by applying the genetic operators (crossover and mutation) until new individuals are obtained.

If *Sk* is the mating pool of the *k* selected chromosomes, by applying the crossover operator, we obtain *Ck* other chromosomes:

Crossover (Sk) = Ck

By applying the mutation operator with the chromosomes *Ck*, we obtain *Gk* new chromosomes:

Mutation (Ck) = Rk

Then, the mathematical model is:

$$X_{n+1} = f(X_n, n)$$
$$f(X_n) = X_n + Rk - Wk \quad (2)$$

Where:

- *Xn* : current population
- *Rk* : new chromosomes from the application of genetic operators
- *Wk* : individual to be eliminated from the current population

3.3 Chromosome Encoding

A chromosome is represented by a two-dimensional array *I (nr x nt)* where nr is the number of classrooms and n_t the number of periods and each element is an object formed by the triplet <teacher-course-class>.

Periods / Classroom	T_1	T_2	⋯	T_{n_t}
R_1	$\langle P_1, M_1, C_1 \rangle$			
R_2		$\langle P_t, M_t, C_t \rangle$		
⋯				
R_{n_r}				$\langle P_k, M_k, C_k \rangle$

Fig -1: The chromosome encoding

3.4 Fitness Function

The value returned by this function is calculated according to the satisfaction of the different constraints. Therefore, for each constraint corresponds one function.

The evaluation function (fitness) is a real function defined by:

$$f: \mathcal{E} \longrightarrow \mathbb{R}^+ \qquad I \longmapsto f(I) = \sum_{i=1}^{14} f_i \quad (3)$$

Where

$$\mathcal{E} = \{\mathcal{E}_C, \mathcal{E}_T, \mathcal{E}_M, \mathcal{E}_\varepsilon, \mathcal{E}_S, \mathcal{E}_R\}$$

and

$$f_i,\ i = \{1, 2, \cdots, 14\}$$

are respectively defined as follow:

$$f_1 = \alpha_1 \sum_{i=1}^{nc} H_1(L_{O_i}) \qquad H1(Loi) = \begin{cases} 1, & if\ Loi \notin I \\ 0, & other\ case \end{cases} \quad (4)$$

Where

- $\alpha 1$ is the penalty point if an object is omitted.
- I : current individual
- Lo : object formed by the triplet <teacher - course – class>
- no : total number of objects

$$f_2 = \alpha_2 \sum_{k=1}^{n_p} \sum_{i=1}^{n_O} \sum_{j=1}^{n_O} H_2(P_k, L_{O_i}, L_{O_j} \qquad H2(Pk, Loi, Loj) = \begin{cases} 1, & if\ for\ Pk, T(Loi) = T(Loj) \\ 0, & other\ case \end{cases} \tag{5}$$

Where

- $\alpha 2$ is the penalty point if a teacher has two classes at the same time
- T (Lo) : period allocated to the Lo object

$$f_3 = \alpha_3 \sum_{k=1}^{n_c} \sum_{i=1}^{n_O} \sum_{j=1}^{n_O} H_3(C_k, L_{O_i}, L_{O_j} \qquad H3(Ck, Loi, Loj) = \begin{cases} 1, & if\ for\ Ck, T(Loi) = T(Loj) \\ 0, & other\ case \end{cases} \tag{6}$$

Where $\alpha 3$ is the penalty point if a class attends two different classes at the same time.

$$f_4 = \alpha_4 \sum_{k=1}^{n_r} \sum_{i=1}^{n_O} \sum_{j=1}^{n_O} H_4(R_k, L_{O_i}, L_{O_j}) \qquad H4(Rk, Loi, Loj) = \begin{cases} 1, & if\ R(Loi) = R(Loj) = Rk \\ 0, & other\ case \end{cases} \tag{7}$$

Where

- $\alpha 4$ is the penalty point if two different classes meet in a room at the same time
- R(LO) : room allocated to the object LO

$$f_5 = \alpha_5 \sum_{i=1}^{n_p} \sum_{j=1}^{n_O} \sum_{k=1}^{n_t} H_5(P_i, L_{O_j}, T_k) \qquad H5(Pi, Loj, Tk) = \begin{cases} 1, & if\ (T(Loi) = Tk) \cap (Tk \notin F(Pi)) \\ 0, & other\ case \end{cases} \tag{1}$$

Where

- $\alpha 5$ is the penalty point if a teacher's subject is placed in the period when he is not available.
- F(P) : availability of teacher P

$$f_6 = \alpha_6 \sum_{i=1}^{n_O} H_6(L_{O_i}) \qquad H6(Loi) = \begin{cases} 1, & if\ R(Loi) = Y(Lo(M)) \\ 0, & other\ case \end{cases} \tag{9}$$

Where

- $\alpha 6$ is the penalty point if the room allocated to a material is not suitable
- Y(LO(M)) : type of room appropriate to matter M of object LO

$$f_7 = \alpha_7 \sum_{i=1}^{n_O} \sum_{j=1}^{n_O} H_7(L_{O_i}, L_{O_j}) \qquad H7(Loi, Loj) = \begin{cases} 1, & if\ (Loi(P) = Loj(P)) \cap (Loi(M) = Loj(M)) \\ 0, & other\ case \end{cases} \quad (10)$$

Where

- $\alpha 7$ is the penalty point if a material of a common core is placed in different periods for all classes.
- Lo(M) : the subject matter of the object Lo
- Lo(P) : the teacher of the object Lo

$$f_8 = \alpha_8 \sum_{i=1}^{n_j} \sum_{j=1}^{n_O} \sum_{j=1}^{n_O} H_8(J_i, L_{O_j}, L_{O_k} \qquad H8(Ji, Loj, Lok) = \begin{cases} 1, & if\ T(Lok/Ji) - T(Loj/Jj) \neq 1 \\ 0, & other\ case \end{cases} \quad (11)$$

Where

- $\alpha 8$ is the penalty point if periods allocated to the same subject in the same day are scattered
- T(Loi/Jj): period attributed to object Lo on days J

$$f_9 = \alpha_9 \sum_{i=1}^{n_c} \sum_{j=1}^{n_j} \sum_{k=1}^{n_O} H_9(C_i, J_j, L_{O_k} \qquad H9(Ci, Jj, Lok) = \begin{cases} 1, & if\ Count\,(Lok/(Ci, Jj)) = 1 \\ 0, & other\ case \end{cases} \quad (12)$$

Where

- $\alpha 9$ is the penalty point if the period number of a one-day class is only one.
- Count(L_{ok}/ (C, J)) : number of objects assigned to class C at days J

$$f_{10} = \alpha_{10} \sum_{i=1}^{n_c} \sum_{j=1}^{n_j} \sum_{k=1}^{n_O} \sum_{l=1}^{n_O} H_{10}(C_i, J_j, L_{O_k}, L_{O_l} \quad (13)$$

$$H10(Ci, Jj, Lok, Lol) = \begin{cases} 1, & if\ T(Lol/(Ci, Jj)) - T(Lok/(Ci, Jj)) \geq emax \\ 0, & other\ case \end{cases}$$

Where

- $\alpha 10$ is the penalty point if there is a large amount of time between classes in a day.
- T(Lol/(C, J)) : period assigned to an object of class C at days J
- e_{max} : maximum number of free time allowed

$$f_8 = \alpha_{11} \sum_{i=1}^{n_c} \sum_{j=1}^{n_j} \sum_{k=1}^{n_O} H_{11}(C_i, J_j, L_{O_k} \qquad H11(Ci, Jj, Lok) = \begin{cases} 1, & if\ Count(Lok/(Ci, Jj)) \geq hmax \\ 0, & other\ case \end{cases} \quad (14)$$

Where

- $\alpha 11$ is the penalty point if the class hours of a one-day class exceed the prescribed standard.
- Count(Lok/(C, J)) : number of objects assigned to class C at day J
- H_{max} : maximum number of hours (periods) of course allowed in a day

$$f_{12} = \alpha_{12} \sum_{i=1}^{n_p} \sum_{j=1}^{n_j} \sum_{k=1}^{n_O} H_{12}(P_i, J_j, L_{O_k} \qquad H12(Pi, Jj, Lok) = \begin{cases} 1, & if\ Count\,(Lok/(Ci, Jj)) \geq tmax \\ 0, & other\ case \end{cases} \tag{15}$$

Where

- $\alpha 12$ is the penalty point if the class hours of a one-day class exceed the prescribed standard.
- Count (L_{ok}/(P, J)) : number of objects attributed to the teacher P at day J
- t_{max} : maximum daily load allowed for a teacher

$$f_{13} = \alpha_{13} \sum_{i=1}^{n_m} \sum_{j=1}^{n_t} \sum_{k=1}^{n_t} H_{13}(M_i, T_j, T_k) \qquad H13(Mi, Tj, Tk) = \begin{cases} 1, & if\ R(Mi/Tj) \neq R(Mi/Tk) \\ 0, & other\ case \end{cases} \tag{16}$$

Where

- $\alpha 13$ is the penalty point if the rooms allocated to a given subject are different.
- R(M / T) : room assigned to the matter M at T period

$$f_{14} = \alpha_{13} \sum_{i=1}^{n_O} \sum_{j=1}^{n_r} H_{14}(L_{O_i}, R_j) \qquad H14(Loi, Rj) = \begin{cases} 1, & if\ C(Ri/Loj(C)) < S(Loj(C)) \\ 0, & other\ case \end{cases} \tag{17}$$

Where

- $\alpha 14$ is the penalty point if the number of students exceeds the room capacity
- C(R/Lo(C)) : room capacity assigned to object Lo
- S(Lo(C)) : number of students affected by object Lo

3.5 Initialization Procedure

The initial population, considered as an initial set of solutions, is formed by *n* chromosomes that are created in a random way. The number *n* remains constant from generation to generation. Since each chromosome is a matrix of dimension *(e * t)* then we have:

$$\mathcal{X}_N = \{I_i[r, t]/i \in [1, n_{pop}], r \in [1, n_r], t \in [1, n_t]\} \tag{18}$$

Algorithm to generate the initial population:

Input (s):

L_O: list of objects

npop: number of individuals

Output (s) :

X_0: set of individuals in the initial population

1: k = 0

2: While $k \leq npop$ do

3: k = k + 1

4: Create the individual I_k: *CreerIndividu(I_k)*

5: Insert I_k in X_0: $X_0 = X_0 + \{I_k\}$

6: End While

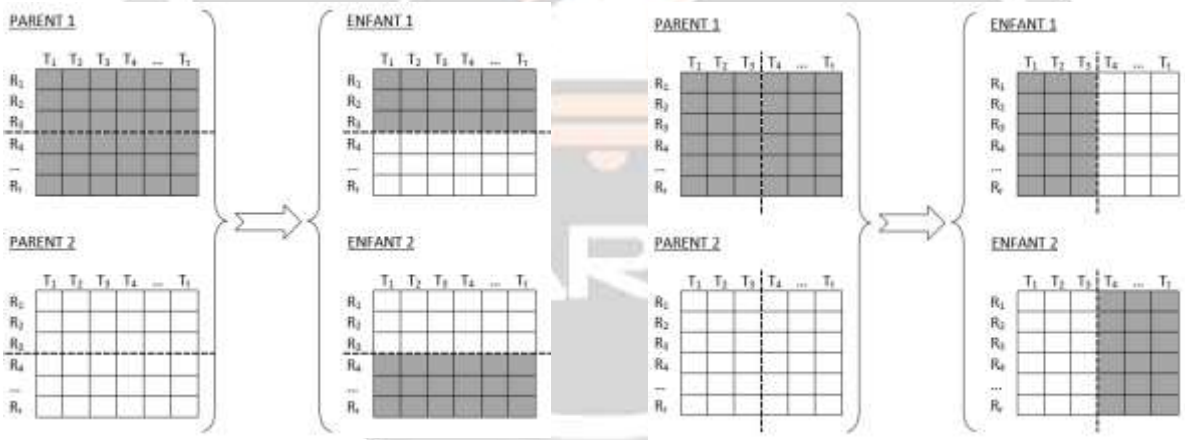
7: Return X_0

3.6 Crossover

The crossover process runs in two steps:

1. Choice and application of crossing probability pc for each individual. Here, there are two options: the first is to pass every individual once and only once to the crossover process and the second allows individuals to participate in a number of times; in this case, the number of new individuals wanted to be specified
2. Crossing process between two parents to give two children

Our coding allows to have horizontal crossing and vertical crossing. The first allows objects to swap rooms and the second allows them to exchange time periods.



(Horizontal crossing) (Vertical crossing)

Fig -2: Illustration of Crossover operator

3.7 Mutation

It is a process where the value of a component of the individual is randomly modified with a low probability of mutation p_m.

In our case, the mutation operator is to choose two allocations and to exchange their contents.

Through the mutation process, an object can change period, room, or both:

- the change of period occurs when the two objects are on the same line
- there is a change of room if they have the same column
- period and room changes are made if they are on different rows and columns

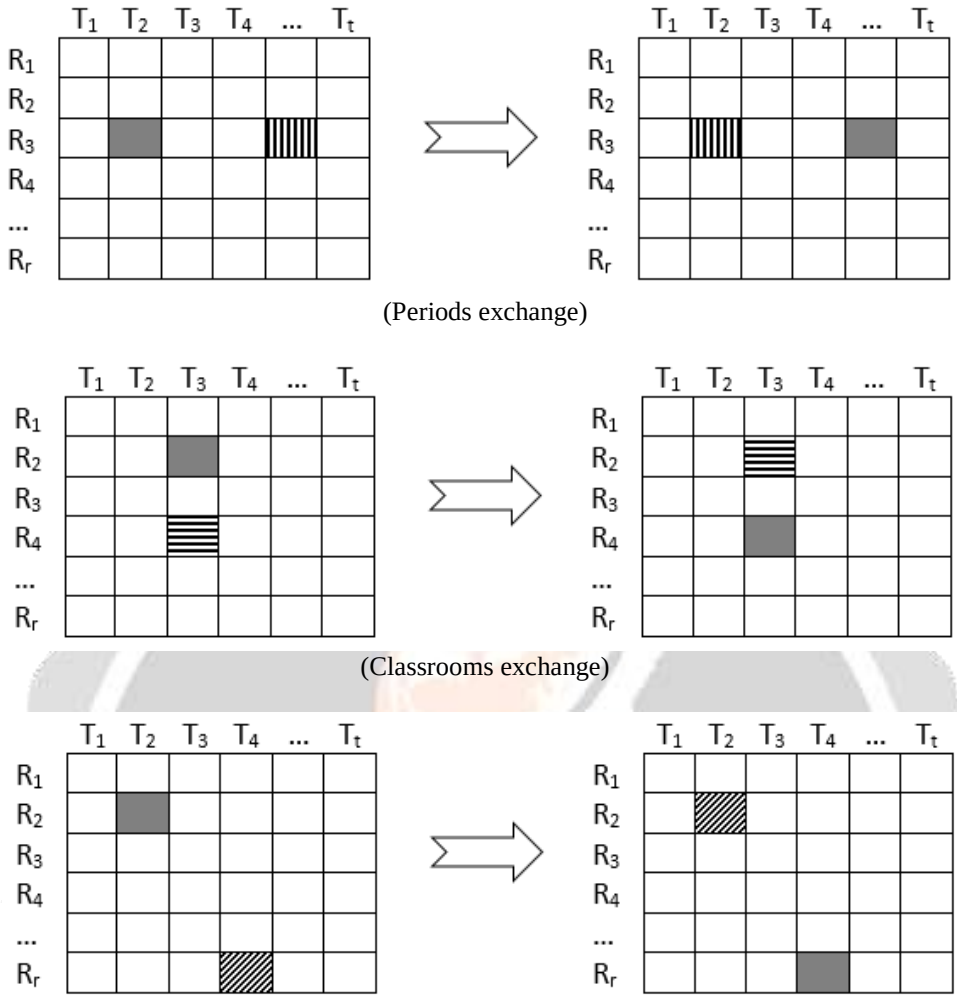


Fig -3: Illustration of Mutation operator

3.8 Replacement

The replacement consists in introducing the descendants obtained by the successive application of the genetic operators in the population of their parents to form the new population and this by constantly keeping the individuals number.

We present three alternative methods of replacement:

- Generational replacement: «replace all parent individuals by individual children»

$$X_{k+1} = \left\{ I_i \in X_{enf},\ i = \{1, 2, \cdots, n_{pop}\} \right\} \tag{19}$$

- The elitist replacement: «keep the best individuals of parent's population and replace the worst elements»

$$X_{k+1} = \left\{ I_i \cup I'_j / I_i \in X_k,\ I'_j \in X_{enf},\ i = \{1, 2, \cdots, n_{prt}\}, j = \{1, 2, \cdots, (n_{pop} - n_{prt})\} \right\} \tag{20}$$

- The replacement by selection: «combine parents and children populations and realize a selection of *npop* individuals»

$$X_{k+1} = \left\{ I_i \in \{X_k \cup X_{enf}\},\ i = \{1, 2, \cdots, n_{pop}\} \right\} \quad (21)$$

3.9 Intelligent operators

The random choice of crossing points risk to reproduce invalid individuals. Intelligent Crossing process define the number and the positions of the crossing points.

In the same way, choose randomly two objects of any individual to be mutated can render it invalid.
Intelligent mutation process is to choose randomly choice the first object and the second one is chosen by applying search criteria based on the firsts parameters.

3.10 Stop Criteria

Generally, a genetic algorithm ends after fixed number of generations. GA can be stopped when a certain condition is reached, for example when the quality of an individual exceeds a certain threshold. For our case, a number of generations is to be specified; failing this, it is the convergence of the population that determines the end of the algorithm.

4. EXPERIMENTAL RESULTS

4.1 Algorithm Complexity

There exist numerous methods to measure the performance of algorithm *A*:

- Runtime: the time spent to find solution *s**
- Memory: the memory space required
- The quality of the solution
- Robustness: the capacity of the algorithm to adapt to problem's input changing

Runtime is dependent on the machine used. The theory of complexity has been introduced to measure the performance of an algorithm without considering the speed of the machine on which it is executed.

A series of tests (**Table -1**), with different data, were conducted to determine the time required for our process to solve the scheduling problem and we can conclude that:

1. The execution time of the algorithm depends entirely on the number of generations G: whatever the size of the population N, this duration remains the same for an equal number of generations
2. The representative curve of these time data is a straight line (**Fig -4**)

So, the time complexity of our algorithm is given by

$$T = a \times G + b$$

which is a polynomial equation of the first degree

Table -1: Results of a set of tests

G \ N	25	50	75	100	150	200
25	84.6834	84.6834	84.6834	84.6834	84.6834	84.6834
50	157.0059	157.0059	157.0059	157.0059	157.0059	157.0059
75	222.2788	222.2788	222.2788	222.2788	222.2788	222.2788
100	292.2515	292.2515	292.2515	292.2515	292.2515	292.2515
125	361.0492	361.0492	361.0492	361.0492	361.0492	361.0492
150	429.8469	429.8469	429.8469	429.8469	429.8469	429.8469
175	498.6447	498.6447	498.6447	498.6447	498.6447	498.6447
200	567.4424	567.4424	567.4424	567.4424	567.4424	567.4424
225	636.2401	636.2401	636.2401	636.2401	636.2401	636.2401
250	716.7182	716.7182	716.7182	716.7182	716.7182	716.7182
275	773.8356	773.8356	773.8356	773.8356	773.8356	773.8356
300	858.6048	858.6048	858.6048	858.6048	858.6048	858.6048
325	926.5443	926.5443	926.5443	926.5443	926.5443	926.5443
350	998.5363	998.5363	998.5363	998.5363	998.5363	998.5363
375	1070.5283	1070.5283	1070.5283	1070.5283	1070.5283	1070.5283
400	1142.5204	1142.5204	1142.5204	1142.5204	1142.5204	1142.5204

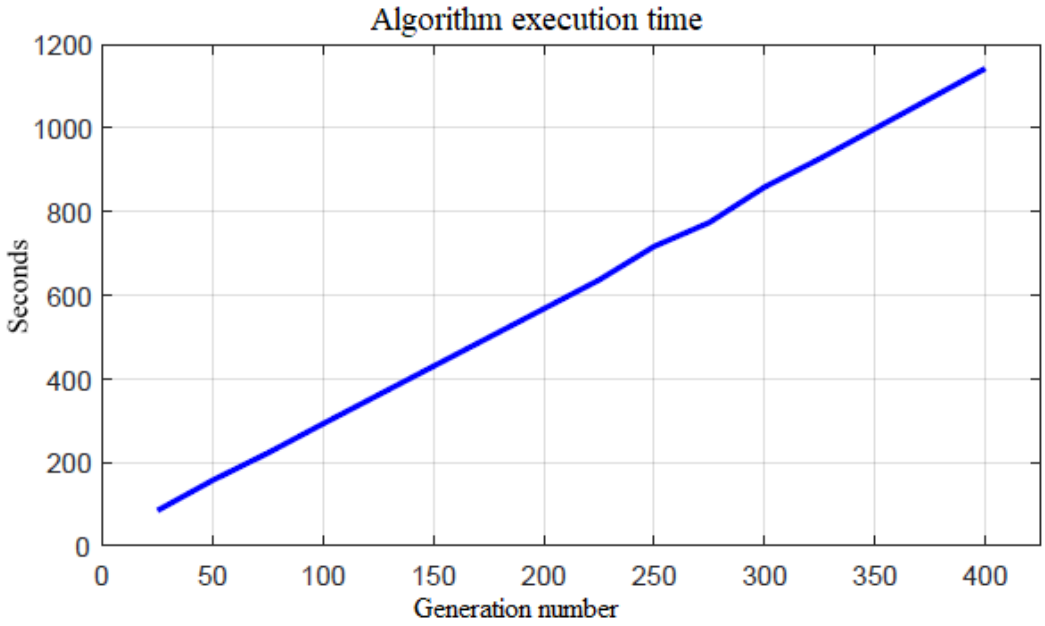


Fig -4: Representative curve of a set of tests

4.2 Fitness Result

We introduced the notion of intelligent operators that consists in optimizing crossover and mutation. The comparative results of standard and intelligent operators are shown in Figure 8, in which we design three curves representing respectively the worst individual, the average individual and the best individual of each generation.

We used the same parameters, the same data and the same generation number and we can see the following major points (generation number G=75):

- Like standard operators, intelligent operators converge too
- The convergence point is reached at half-time compared to standard operators.
- The intelligent operators improve the convergence point value

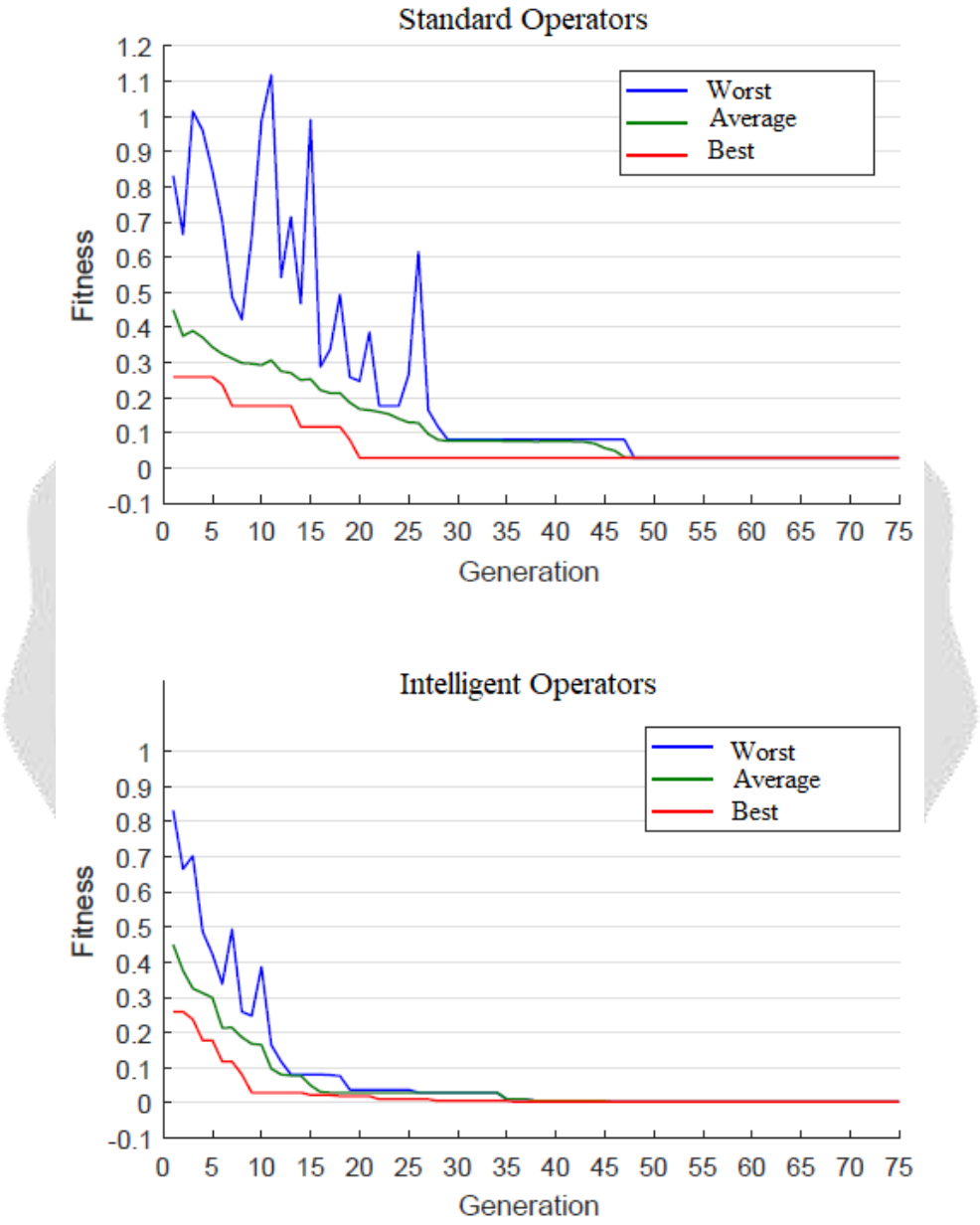


Fig -5: Fitness results

Table -2 summarizes the comparative results of standard and intelligent operators. The advantages of these are highlighted

- Execution time: the duration of the resolution process is reduced
- Quality of the solution: approximation to thousandth-close to the optimal value

Table -2: Results of a set of tests

Operator		Standard	Intelligent
Individual fitness at the beginning	Worst	13.1681	13.1681
	Average	13.5502	13.5502
	Best	13.7410	13.7410
Individual fitness at the end	Worst	13.9614	13.9975
	Average	13.9614	13.9975
	Best	13.9614	13.9975
Convergence at the generation		47	37

5. CONCLUSIONS AND FUTURE WORKS

We have been able to find a mathematical model and how to implement it with genetic algorithms approach. The scheduling problem is NP - hard, the efficiency of our approach is that its complexity is polynomial ($\mathcal{O}$ (n)), it means that the algorithm runs in reasonable time. Furthermore, fitness function has a tendency to zero, which leads to the optimal solution at a certain generation.

However perfectly we have achieved the goal, many situations remain to be discussed. We have limited ourselves to taking up the key elements of teachers' cases. The study can be extended by considering other cases such as the student choices.

This research can be used to resolve all scheduling problems in the academic field such as "Course Scheduling System", "Timetable Examination", "Maximize Participant Satisfaction". It can also use to solve all types of optimization problems such as the TSP (Traveling Salesman Problem), the NSP (Nurse Scheduling Problem), the VRP (Vehicle Routing Problem), the FCT (Fixed Charge Transportation), the BAP (Berth Allocation Problem) and OSS (Open Shop Scheduling).

6. REFERENCES

[1] MirHassani, S.A., A computational approach to enhancing course time-tabling with integer programming, *Applied Mathematics and Computation* 175(1), 2006. Pp.814-822.

[2] Wren, A., Scheduling, Timetabling and Rostering–A special Relationship, *In: Burke, E. and Ross, P. (Eds.), Practice and Theory of Automated Timetabling*, 1996, pp.46-75.

[3] Burke, E.K., Jackson, K., Kingston, J. and Weare, R., Automated university timetabling: The state of the art. *Comput J* 40, 1997, pp.565-571.

[4] Burke, E.K. and Petrovic, S., Recent research directions in automated timetabling. *Eur J Opl Res* 140, 2002, pp. 266-280.

[5] Chaudhuri A, De K., Fuzzy Genetic Heuristic for University Course Timetable Problem. *IJASCA*, Vol.2, No.1, 2010.

[6] Syahputra, M.F., Apriani, R., Sawaluddin, Abdullah, D., Albra, W., Heikal, M., Abdurrahman, A., Khaddafi, M., Genetic algorithm to solve the problems of lectures and practicums scheduling. *IOP Conf. Series: Materials Science and Engineering* 308, 2018.

[7] Ten Eikelder. Some complexity aspects of secondary school timetabling problems. *Lecture notes in computer science*, vol. 2079, 2001, pp. 18-27.

[8] Sanjay R. Sutar and Rajan S. Bichkar. High School Timetabling using Tabu Search and Partial Feasibility Preserving Genetic Algorithm, *International Journal of Advances in Engineering & Technology*, 2017.
[9] Mamede, N., Rente, T., Repairing Timetables using Genetic Algorithms and Simulated Annealing, *In Burke, E. and Carter, M. (Eds.): The Practice and Theory of Automated Timetabling II*. 1997.
[10] Goldberg, D.E., The design of innovation: lessons from genetic algorithms, lessons for the real world. *IlliGAL Report* No. 98004, 1998.