Palestine Polytechnic University
College of Administrative Science and Informatics
Information Technology Department

University Course Scheduling Using Evolutionary Algorithms

Prepared by
Safa' Ibrahim Adi
Mohammad Khalil Abu Qopita

Supervised by
Dr. Mohammed Aldasht.

12/7/2008

Project submitted in partial fulfillment of the requirements for the degree B.Sc. in information technology in Palestine polytechnic university

June – 2008

الملخص

يقوم هذا البحث على ايجاد حل مناسب في فضاء مشكلة جدولة مواعيد محاضرات الجامعة عن طريق تصميم خورازمية تطورية (Evolutionary algorithms) تراعي الامور المتبعة في حل المشكلة.

حيث يقوم البحث بايجاد برنامج لمواد الجامعة بحيث يراعي القيود الموضوعة بنوعيها (hard, soft) ، فينتج لدينا تعيين لشعب المساقات في وقت محدد وقاعة مناسبة بحيث لا يوجد اي تعارض زمني مع اي مادة اخرى ،حيث تم تطبيق هذا البحث على البيانات الحقيقية لكلية العلوم الادارية ونظم المعلومات.

لقد بدانا المشروع بتعريف المشكلة ثم تحديد القيود الواجب مراعاتها، بعدها تم انشاء وصف النموذج الرياضي لحل المشكلة، واخيرا تم بناء خوارزمية تعمل على البحث في البيانات لايجاد حل مناسب لهذه المشكلة.

وقد تمكنا من الحصول على نتائج جيدة مقارنة مع البرامج التي توضع يدويا من قبل الجامعة. حيث تمكنا بالمعدل أن نحقق (soft constraints) بنسبة 97%. أما (hard constraints) تم تحقيقها بنسبة 100% تقريبا.

تبين النتائج أن الحل المقترح من خلال البرمجية التطورية التي تم بنائها عمل على حل مشاكل التسجيل التي كانت لدى أكثر من ثلث الطلاب بسبب الجدول الموضوع يدوياً حيث تلبي رغبة الطلاب بتسجيل عدد الساعات التي كانوا يرغبون بها على عكس الجدول الموضوع يدوياً، وبذلك لقي الجدول المقترح نسبة قبول أعلى بكثير لدى الطلبة من تلك التي تتعلق بالجدول الموضوع يدوياً.

Abstract

Evolutionary algorithms provide mechanisms that can achieve an efficient exploration for design spaces. Thus, they constitute a powerful tool for identifying the best alternatives to implement the solution of a certain problem. In this work we apply the evolutionary algorithms to the university course scheduling problem, where a feasible and comfort time tables are required. This means a suitable assignment of course sections to a classroom or lab at a certain time. In general, this is a search problem for a minimum-cost solution taking into account a set of hard and soft constraints. Then, we apply the proposed methodology on a real data taken from the Collage of Administrative Sciences and Informatics at Palestine Polytechnic University.

In the present work we start defining the problem and determining the constraints under which we the solution should be found. In the second step, we explain the problem model, which defines the sets that are used in the implementation, such as courses, rooms, instructors, and students groups. Finally, an evolutionary search method is applied to the implemented university course scheduling framework.

The proposed methodology is applied on a real data set from the collage of administrative sciences and informatics at the Palestine polytechnic university. With all its complexities and constraints. Results show that our methodology can give better time schedule than those performed manually. In average, soft constrains were fulfilled up to 97% and hard constrains were satisfied up to 100%.

The obtained results show that the proposed solutions can solve many registration problems as to permit the students to register what they want.

Acknowledgment

We would like to express our deep thanks and appreciation to our mentor and teacher Dr. Mohammed Aldasht, who provided us with the needed knowledge and supported us step by step, to complete the project.

We are grateful to those people who care the most about us…

To those who own the warmest hearts ever…

To the third partner in our project…

We are grateful to our loving…

Parents.

Love you for always and for ever.

We are grateful to those people who we love…

Brothers, sisters and friends.

We would like to express our deep thanks to our brother Fathi Mohammed Salhab.

We would like to thank our teacher Mr. Akram Ihshiash.

Contents

List of Figures

List of Tables

Chapter One: Introduction

1.2 Overview.

1.2 Scheduling problem.

1.3 Related works.

1.1 Overview

University course scheduling can be considered as an instant of what so called timetabling problem. In which, timeslots and teachers must be assigned to a set of courses in a way that satisfies a set of hard constraints and minimize the cost of a set of soft constraints [1]. This problem is considered to be NP-Complete, or even NP-hard problem; this implies that there is no known polynomial time algorithm that can guarantee finding a feasible solution [2].

A problem H is NP-hard (nondeterministic polynomial-time hard) if and only if there is an NP-complete problem L that is polynomial time Turing-reducible to H, NP-hard problems may be of any type: decision problems, search problems, optimization problems [15].

1.2 Scheduling problem

Scheduling problem, in general, can be defined as a problem of finding the optimal sequence for executing a finite set of operations under a set of a certain constraints [3]. Scheduling problem has several types, such as: multiprocessor scheduling, shop scheduling, staff scheduling, timetable design, etc. In this work, the university course scheduling problem will be handled; this includes, the process of assigning a set of classrooms and set of instructors to a given set of courses taking into account a set of constraints that may be hard or soft ones.

Hard constraints like no person can be in more than one place at a time, and the total resources allocated to some time slot must be less than or equal to the resources that are available in that period, etc. On the other hand, soft constraints like some teacher should not have very late classes daily, a group of students should not have consecutive classes in different and large distant places, and other individuals preferences, etc. [4] [5].

1.3 Related works

There are many works proposed in the literature to solve such problem. In [1], two approaches are proposed to solve the problem, modified genetic algorithm (MGA) and cooperative genetic algorithm (CGA). Results show that (MGA) method was

significantly enhanced algorithm performance with modified basic genetic operators; In addition, algorithm performance is considerably improved by using cooperative genetic method. In [6], they report on the development of a general tool called NExshed, this tool is implemented as a plug-in for Microsoft excel. In [7] they used a hybrid algorithm for solving the timetabling problem which is commonly encountered in many universities, this algorithm combines an integer programming approach, a greedy heuristic, and a modified simulated annealing algorithm collaboratively is proposed to solve the problem. In [8], a method is presented to translate the course requirements into a relational formula, thus allowing the scheduling problem to be solved with existing tools such as the Allay Analyzer. In [9], they address academic course scheduling in a networked environment using intelligent agents within a decision support framework. In [10], they have investigated a variety of approaches based on simulated annealing, including mean-field annealing, simulated annealing with three different cooling schedules, and the use of a rule-based preprocessor to provide a good initial solution for annealing. In [3], a multi-agent system for university course timetable scheduling is proposed. In [4], they improved a complete approach using integer linear programming (ILP) to solve the problem. The (ILP) model is developed and solved using the three advanced ILP solvers based on genetic algorithm and Boolean satisfiability (SAT) techniques.

As mentioned above, many works are proposed to solve the problem, but each one model the problem according to a set of parameters defined to be suitable for his institution, and different from each other. Because of that we need to assess the problem depending on our needs and taking into account our private rules and conditions related to the timetable put on the start of each semester, especially, the variations in set of soft constraints, which will decide the fitness of the solution.

Chapter Two: Heuristics for solving NP-hard problem

3.1 Evolutionary Computation.

3.2 SWRM intelligence.

2.1 Evolutionary Computation

Evolutionary computation is a subfield of computational intelligence which provides mechanisms that can achieve efficient exploration for design spaces. Thus, evolutionary algorithms constitute an efficient tool for identifying the best alternatives to implement the solution of a certain problem. They use an iterative progress method, similar to the growth or development in a population. This population is then selected in a *guided random* (stochastic) parallel to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution.

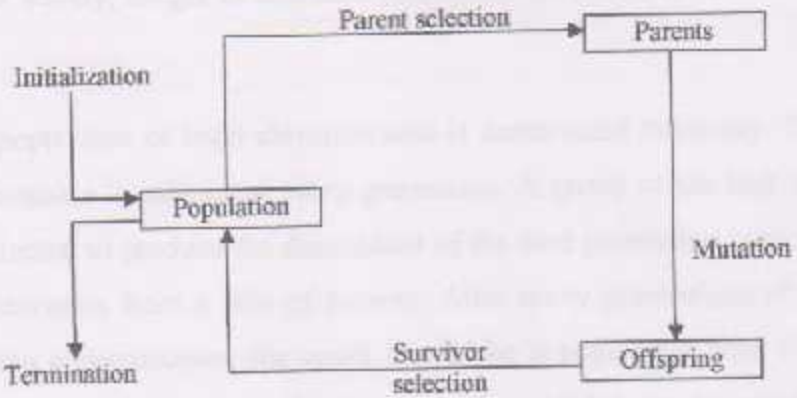


Figure (2.1): General Scheme of EAs: *From "A.E. Eiben and J.E. Smith, Introduction to Evolutionary"*

Evolutionary computation can be classified into four main areas:

1- Genetic algorithms.

2- Evolution strategies.

3- Genetic programming.

4- Evolutionary programming.

1- Genetic algorithms

Genetic algorithms were invented by John H. Holland to follow the natural selection and evolution process. In the nature, each species has to adapt itself according to a complex environmental conditions and changes with the purpose of maximizing the probability to survive. The knowledge collected by each species is coded in chromosomes, which go through transformations when they are reproduced.

During a period of time those changes on the chromosomes will reproduce a species with greater probability of survive, and has a greater probability of passing its

enhanced characteristics to future generations. Surly, not all changes are beneficial, thus, those whose changes are not beneficial tend to die.

Holland's genetic algorithm tries to simulate the natural selection and evolution in the following way. The first step is the solution representation to be legal with the considered problem by a sequence of genes which can have a value from a specific finite variety. This sequence of genes that represents a solution is called a chromosome. The most popular way of solution coding is the binary representation of the solution. Rarely, integer or character representation can be used.

An initial population of legal chromosomes is constructed randomly. The fitness of each chromosome is calculated every generation. A group of the best chromosomes are then selected to produce the descendant of the next generation, which inherits the best characteristics from a pair of parents. After many generations of selecting the most suitable chromosomes, the result should be a population with a considerably better fitness than the original. Genetic algorithms (GAs) can be considered as and optimization techniques based on the concepts of natural and genetic selection [11, 12].

When the genetic algorithm is implemented it is done in a manner that involves the following cycle:

- Evaluate the fitness of all of the individuals in the initial (parent) population.
- Create new population by performing operations such as selection, crossover, and mutation on the individuals.
- Discard the old population and iterate using the new population.

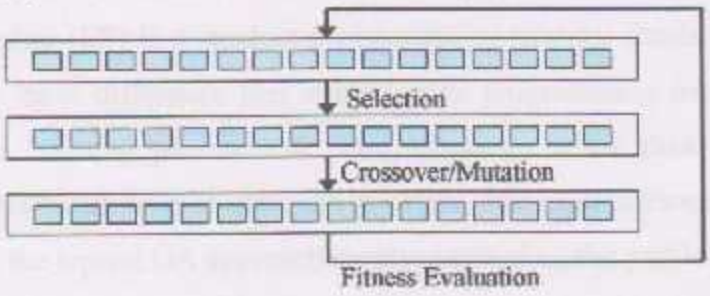


Figure (2.2): Genetic Algorithm cycle.

One iteration of this loop is referred to as a generation, the first generation (generation 0) of this process operates on a population of randomly generated individuals, and for this reason the algorithm is concerned with the fitness to improve the population.

2- Evolution strategies

Evolution strategy is an optimization technique, Developed by Rechenberg (1965, 1973) and Schwefel (1965, 1977) at the Technical University of Berlin, based on ideas of adaptation and evolution, which use mutation, recombination, and selection applied to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions. ESs have the following basic characteristics:

- ESs emphasize mutation and recombination as essential operators.
- The selection operator is deterministic.
- Parent and offspring population sizes usually differ from each other.

3- Genetic programming

Genetic programming is the extension of the genetic model of learning into the space of programs. It applies the evolutionary search principle to automatically develop computer programs in suitable languages (often LISP, but others are possible as well), genetic programming breeds computer programs. That means the objects that establish the population are not fixed-length character strings that encode possible solutions to the problem, but they are programs that when executed are the candidate solutions to the problem.

4- Evolutionary Programming

Evolution programming (EP) is a stochastic optimization strategy similar to genetic algorithms, but the main difference that evolutionary programming insists on the behavioural linkage between parents and their offspring, rather than seeking to emulate specific genetic operators as observed in nature. The main difference between EP and GAs is that, the typical GA approach involves encoding the problem solutions as a string of representative signs (binary representation). In EP, the representation comes from the nature of the problem. Furthermore EP highly depends on mutation

operation. Mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behaviour of the offspring as highly probable and substantial variations. Further, the intensity of mutations is often reduced as the global optimum is approached. A special concern with the mutation, when the global optimum is not already known, several techniques have been proposed and implemented which address this difficulty, the most widely studied being the "Meta-Evolutionary" technique in which the variance of the mutation distribution is subject to mutation by a fixed variance mutation operator and evolves along with the solution.

Evolutionary programming is the more flexible approach to evolution than some of the other techniques. Generally it depends on mutation process and not on the recombination like (cross over) to produce offspring.

Evolutionary programming process begins with selecting parents from population to reproduce; there properties are mutated to produce children who are added to population, which doubled the population size. Then discards unsuitable individuals and selects the best individuals from the population to bring it back to the first size in order to use it in the next generation.

Evolutionary programming method consists of the following steps (Repeat until a threshold for iteration is exceeded or an adequate solution is obtained):

1. Initialize the population.
2. Expose the population to the environment.
3. Calculate fitness for each member.
4. Randomly mutate each "parent" population member.
5. Evaluate parents and children.
6. Select members of new population.
7. Go to step 2 until some condition is met.

Step 1: In the initialization step of this process we initialize the needed data from different files which are: student group's time slots, teacher's time slots, and room's time slots, the algorithm will take knowledge from the mentioned files about the time availability for students, teachers and rooms respectively. Initially it is supposed that

students are available all the time along the five working days of the week, while teachers and rooms may have some time availability constraints.

After data initialization, an initial population is constructed according to the initial data and constraints of the algorithm. Producing the first population. Then we depend on this population in going through the algorithm iterations.

Step 2: fitness calculation in which the calculation of each individual's cost is essential. Population cost is measured as the sum of the cost for each gene in the individual. The cost is determined by scanning the genes of the individual orderly about the violated constraints (hard and soft ones) by every gene; for example a hard constraint violation may happen if, in some gene, a course section is assigned in an occupied time slot for the teacher, room, or a student group. On the other hand a soft constraint violation may happen if a course section is a assigned in a time slot that makes long time wait, or too late, for the teacher of the course or the students.

Step 3: the fitness is calculated for the individual x by the simple function $f(x)=1/(\text{cost}(x)+1)$. The value of this function is important in determining whether this individual will pass to next step (new population) or not.

Step 4: is mutation, where a random modification is done on the genes in the individual; this change can be done with a probability. In our algorithm every individual in population is ordered according to the gene's cost, where genes of zero are put at the beginning of the individual then the mutation starts from the first gene whose cost is greater than zero.

The next to do is the selection of the individuals of the next population depending on the calculated fitness values. Many selection strategies can be implemented in the evolutionary program to determine which individuals to pass to the next generation, here we list below some of the most common methods:

1. Elitist selection: with this strategy the most fit members of each generation are guaranteed to be selected. Pure elitism may conduct the algorithm to rapidly converge to a local minimum. Most evolution programming does not use pure *elitism, but instead use a modified form where the single* best or a few of the best

individuals from each generation are copied into the next generation just in case nothing better turns up.

2. Scaling selection: As the average fitness of the population increases, the strength of the selective pressure also increases and the fitness function becomes more discriminating. This method can be helpful in making the best selection later on when all individuals have relatively high fitness and only small differences in fitness distinguish one from another.
3. Rank selection: Each individual in the population is assigned a numerical rank based on fitness, and selection is based on this ranking rather than absolute difference in fitness. The advantage of this method is that it can prevent very fit individuals from gaining dominance early at the expense of less fit ones, which would reduce the population's genetic diversity and might hinder attempts to find an acceptable solution.
4. Hierarchical selection: Individuals go through multiple rounds of selection each generation. Lower-level evaluations are faster and less discriminating, while those that survive to higher levels are evaluated more rigorously. The advantage of this method is that it reduces overall computation time by using faster, less selective evaluation to weed out the majority of individuals that show little or no promise, and only subjecting those who survive this initial test to more rigorous and more computationally expensive fitness evaluation.
5. Roulette-wheel selection: A form of fitness-proportionate selection in which the chance of an individual's being selected is proportional to the amount by which its fitness is greater or less than its competitors' fitness.

In our algorithm we have used the roulette-wheel selection where the conceptualization is that of a wheel whose surface is subdivided into wedges representing the probabilities for each individual, see Figure (3.3). For instance, one point on the edge is determined to be the zero point, and each arc around the circle corresponds to an area on the number line between zero and one. A random number is generated, between 0.0 and 1.0, and the individual whose wedge contains that number is chosen. In this way, individuals with greater fitness are more likely to be chosen. The selection algorithm can be repeated until the desired number of individuals has been selected.

In roulette wheel selection, the probability of an individual being selected (the size of its slice of the wheel) is proportional to its fitness (indicated here by shading).

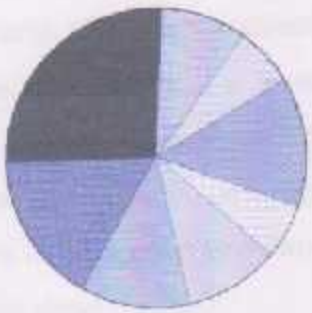


Figure (2.3): Roulette wheel selection

2. Flowchart of evolutionary programming

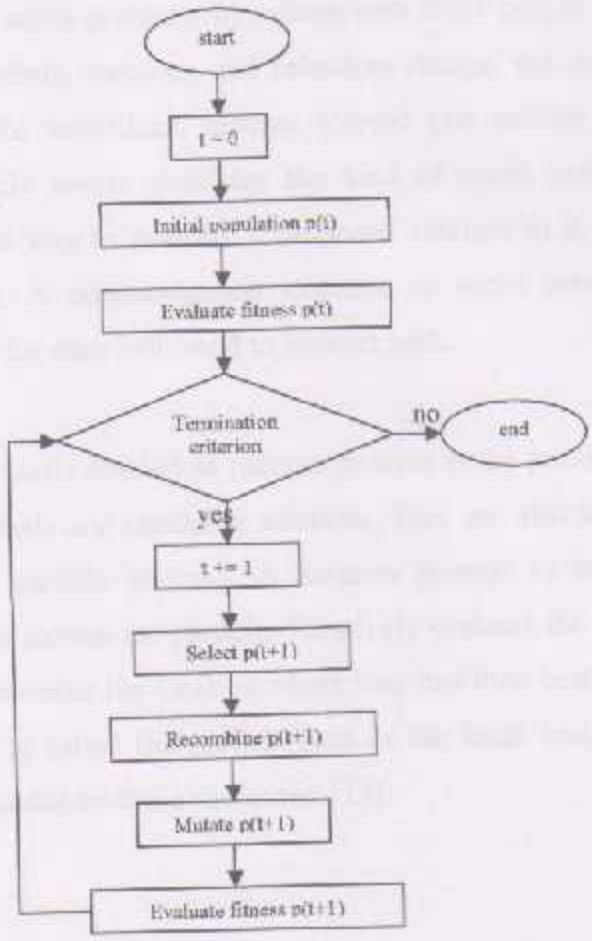


Figure (2.4): The flowchart of the evolutionary algorithm.

2.2 SWRM intelligence

A swarm is a population of interacting elements that is able to optimize some global objective through collaborative search of a space. Interactions that are relatively local (topologically) are often emphasized. There is a general stochastic (or chaotic) tendency in a swarm for individuals to move toward a center of mass in the population on critical dimensions, resulting in convergence on an optimum. This strategy may be applied in a future work in order to enhance the final results. The next section contained more explanation about the PSO.

PSO algorithm is a swarm intelligence based algorithm to find a solution to an optimization problem in a search space, or model and predict social behavior in the presence of objects. People solve problems by talking with other people about them, and as they interact their beliefs, attitudes, and behaviors change; the changes could typically be depicted as the individuals moving toward one another in a socio cognitive space. The particle swarm simulates this kind of social optimization. A problem is given, and some way to evaluate a proposed solution to it exists in the form of a fitness function. A communication structure or social network is also defined assigning neighbors for each individual to interact with.

Then a population of individuals defined as random guesses at the problem solutions is initialized, these individuals are candidate solutions, they are also known as the particles hence the name particle swarm. An iterative process to improve these candidate solutions is set in motion the particles iteratively evaluate the fitness of the candidate solutions and remember the location where they had their best success. The individual's best solution is called the particle best or the local best. Movements through search space are guided by these successes. [14]

Chapter Three: Problem modeling

2.1 Problem description and modeling.

2.2 Solution definition.

3.1 Problem description and modeling

In this project we will start working on the timetable design problem of one of the four collages of the university. Then, the solution will be generalized on the rest collages. The college of administrative sciences and informatics in Palestine polytechnic university will be considered. In this college there are four academic 4-years programs, which are: information technology (IT), information systems (IS), graphics and multimedia (G), administrative management (M). So, courses are offered at the beginning of each semester for 4 student groups in each academic program, so we have 16 total student groups.

On the other hand, we have 5 work days a week (Sunday to Thursday). On Sunday, Tuesday and Thursday there are nine 60-minutes time slots a day, and on Monday and Wednesday there are six 90-minutes time slots a day.

To handle the problem correctly, we have defined the problem be six different sets: students, instructors, courses, classrooms, time slots (lectures), and the set of constraints. Then, the problem is formulated as following: problem = {S, T, C, R, L, O}. Where:S= $\{s_1, s_2, \ldots, s_i\}$ is the set of student groups, each element in S is a vector (m,y) where m: is the major (IT, IS, G, or M) and y: is the group number which will be represented by academic year of that group, this vector determines a group of students of the same major and academic year. So, for the considered collage, m ranges from 1 to 4, and y ranges from 1 to 4. Thus, S= {(1, 1), (1, 2), …, (4, 3), (4, 4)}.

T = $\{t_1, t_2, \ldots, t_j\}$ is the set of instructors in the college at the current semester, each element in T is a vector (inst_no, inst_cons) where inst_no: is the number of a specific instructor, inst_cons: is an array indicating to timeslots when the instructor can give a lecture.

C = $\{c_1, c_2, \ldots\ldots, c_n\}$ is the set of courses offered for the student in the current semester, each element in C is a vector (s_i, co_no, section, p, t_j, th, ph, ph_type, ph_inst) where s_i: is the group of students, which this course is offered for, co_no: is the course number, section: is the section number of this course, p: is the maximum

capacity of the section, t_j: is the instructor of this section, th: is the number of theoretical hours of this course, ph: is the number of practical hours of this course, ph_type: is the type of the practical hour it is set to 0:if the course has no practical hour,1: if this hour is lab, set to 2 if this hour is multimedia lab, set to 3 if this hour is photo lab, and set to 4 to indicate that this hour is workshop, ph_inst: the period that the teacher must be in the lab it is set to 0: no hour in the lab, 1: half of the lab period, 2: all the lab period.

$R = \{r_1, r_2, \ldots, r_m\}$ is the set of classrooms, each element in R is a vector (c, d, l) where c: is the maximum capacity of this room (number of students can be assigned to this classroom), d: is a flag set to 1 if this room has data show, l: is a flag used to determine the type of this classroom, set to 0 to indicate that the classroom is a room, set to 1 to indicate that the classroom is PC lab, set to 2 to indicate that the classroom is a multimedia lab, set to 3 to indicate that the classroom is photo lab, and set to 4 to indicate that the classroom is workshop.

$L = \{l_1, l_2, \ldots, l_k\}$ is the set of time slots of the week, each element in L is a vector (n,d) where n: is the time slot number, and ranges from 1 to 9 for days 1, 3, and 5, and ranges from 1 to 6 for days 2 and 4, and d is the day ranges from 1 to 5. Thus, L = { (1,1), (2,1), …, (9,1), (1,2), (2,2), … (6,2), …, (8,5), (9,5)} where (1,1) means: the first lecture in the first working day (Sunday), and so on. Each lecture (theoretical hour) on 1, 3, and 5 is 60 minutes, so the workable hour is 3 timeslots (3 hours) in these days. On 2 and 4 each lecture (theoretical hour) is 90 minutes, so the workable hour is 2 timeslots (3 hours) in these days. For example if the course has 3 theoretical hours then it takes 3 timeslots in days 1, 3, and 5 or 2 timeslots in days 2 and 4. Except on 1 and 5 the fifth lecture is 2 timeslots because on 3 there is no fifth lecture.

$O = \{o_1, o_2, \ldots, o_q\}$ is the set of constraints, where o_q: is the penalty weight (cost) of this constraint. Hard constraints will be assigned infinity cost, while soft constraints will assigned some constant cost to indicate the weight of violating that each one.

The sets described above are used to define the problem; on the other hand the set of solutions of such problem is very large set. Each solution will be evaluated using an

indicator that determines how much this solution is good (fitness). Our methodology will use a Evolutionary algorithm to search for some optimal solution.

3.2 Solution definition

$G = \{g_1, g_2, \ldots, g_w\}$ is the set of genes which constitute the individual (chromosome), each gene is a vector (*course, room_no, ph_tslot, lab_no, th_tslot*), look to figure (2.1). where: *course* is the number of course section, *room_no* is the number of room assigned to this course section during theoretical hours, *th_tslot* is timeslot assigned to the course section for theoretical hours, *lab_no* is the number of lab assigned to the course during practical hours and *ph_tslot* is the timeslot (lab) assigned to the course during practical hours.

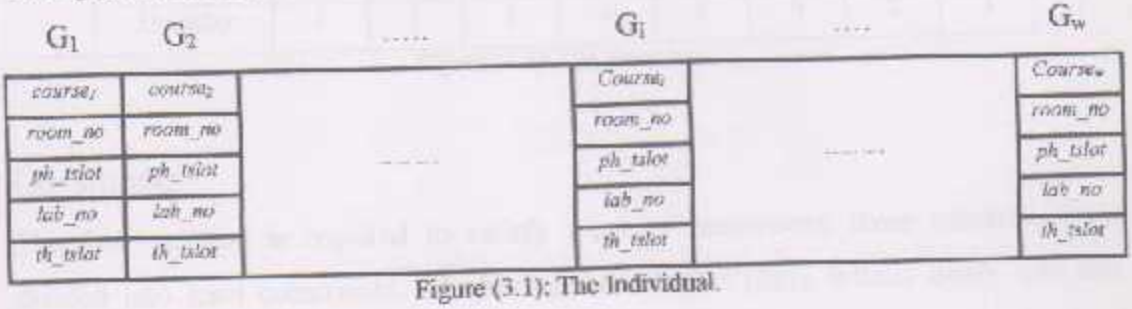


Figure (3.1): The Individual.

We have a complex search domain, where, for each course section, suitable classroom and/or lab are chosen and time slots are assigned such that they will be suitable for the student group and the lecturer. In order to reduce the search time and complexity, our method is randomly chose the suitable place and time for the course section so that the lecturer has no time conflicts, and the search is dedicated for optimizing the solution considering the hard and soft constraints for the students and only the soft constraints for the lecturers.

The individual is constructed in a way that every gene has different number of options when choosing timeslot and classroom; such that, course section in gene G_1 has a maximum number options available when randomly selecting classrooms and timeslots, course section in gene G_2 has 1 option less than those was available for gene G_1, and the last one, gene G_w has the least number of option.

The instance chosen for our application has a maximum of 14 classrooms and 7 labs available. On the other hand, along the week, we have 41 theoretical classes (1 hour each) and 13 practical classes or labs (3 hours each). That means we have a maximum capacity of (41*14) = 574 theoretical classes, and (13*7)= 91 practical classes along the week. As shown in figure (2.2), where 1 means time slot is available, and 0 means time slot is not available.

Days	Time slots								
	Sunday	1	1	1	1	1	0	1	1	1
	Monday	1	1	1	1	1	1	1	1	1
	Tuesday	1	1	1	1	0	0	1	1	1
	Wednesday	1	1	1	1	1	1	1	1	1
	Thursday	1	1	1	1	1	0	1	1	1

Figure (3.2): The timeslots.

Constraints

The final solution is required to satisfy a set of constraints, these constraints are divided into hard constraints, which must be satisfied (have infinity cost), and soft constraints which should be satisfied (have a reduced cost compared to hard constraints). The set of constraints and their weights are as follow:

Hard constraints are:

1. An instructor must not have more than one class at any given time slot. (Violation cost = infinity).
2. An instructor can be assigned classes only in his available time, like part timers. (Violation cost = infinity).
3. A classroom must not assigned more than one class at any given time slot. (Violation cost = infinity).
4. Number of students in any lecture should be less than or equal the maximum capacity of the classroom. (The cost increases as the number of students above the capacity of classroom increases, each student above the capacity increases the cost by a constant until the number of students above capacity is greater than a given small threshold, then the cost of this constraint will equal to infinity).

5. At any given time slot no student group can have more than one class. (Violation cost = infinity).

Soft constraints (violation cost will be constant relative to the constraint importance) are:

1. Students should not have more than three classes consecutively.
2. Instructors should not have long free time between lectures.
3. Instructors should not have very late classes daily.
4. A group of students should not have consecutive classes in different and large distant places.
5. Students should not have long free time between lectures.

Chapter Four: Implementation

4.1 Timetabling system.

4.2 Flowcharts.

4.1 Timetabling system

In this section we talk about the implementation for the project that used to solve the problem which is considered to be one of the NP-hard problems that need intelligent algorithms and parallel computing to find an optimal solution to these problems.

As we mentioned in a previous section, the problem of university course scheduling is divided into sets, the implementation of each set is as follow:

Each element in rooms (R) set is a struct that can be described as:

Name	Data type	Description.
ro_capacity	int	The maximum capacity of this classroom.
d_sh	short	A flag that set to 0:the room hasn't dataShow, 1:the room has dataShow.
type	short	A flag that set to 0:room, 1:PC lab, 2:multimedia lab, 3:photo lab, 4:Workshop.
ts[5][9]	int	The time slots for the classroom.

Table (4.1): The struct of classroom (Rooms).

Each element in students groups (S) set is a struct that can be described as:

Name	Data type	Description
major	int	A flag that set to 0:IT, 1:IS, 2:G, 3:M.
year	int	2000, 2001, 2002, 2003, 2004,.....
ts[5][9]	int	The time slots for the student group.

Table (4.2): The struct of student group (StdGroups).

Each element in instructors (T) set is a struct that can be described as:

Name	Data type	Description
No	int	The number of this instructor.
ts[5][9]	int	The time slots for the Instructor.

Table (4.3): The struct of teacher (Instructors).

Each element in courses (C) set is a struct that can be described as:

Name	Data type	Description
st_id	short	The students group number.
c_no	Int	Course number.
section	short	Section number of this course.
capacity	Int	The maximum capacity of this course.
inst_no	int	The number of this instructor.
th	short	Number of theoretical class hours for this course.
ph	short	Number of practical hours for this course.
ph_type	short	A flag that set to 0:no practical hours, 1:PC lab, 2:multimedia lab, 3:photo lab, 4:Workshop.
ph_inst	short	The period teacher in this lab, 0: 0 hours,1: 1.5 hour, 2 : 3hours.

Table (4.4): The struct of course (Courses).

Each element in individuals (G) set is a struct that can be described as:

Name	Data type	Description
course	int	Course number.
room_no	int	Classroom number.
th_tslot	*int*	*Timeslot for theoretical hours.*
lab_no		lab number.
ph_type	int	Timeslot for practical hours.
cost	long	The penalty of this gene.

Table (4.5): The struct of gene (Gene).

4.2 Flowcharts

This section presents the implementation of the evolutionary algorithm to solve the problem. The number of courses sections stored in a parameter celled *numOfGenes* which determines how many genes will be in each individual (chromosome), then we built a dynamic array of struct for rooms, students groups, instructors, and courses. The number of individuals on each population is defined at the beginning of the program as PopSize.

After that we built the evolutionary algorithm and its functions. We define a variable called *NumGener* which determines the total iterations of the evolutionary algorithm as shown in the following algorithm:

Initialize Population; // creates initial (parent) population.

t = 0;

while (t < NumGener-1)

begin

Population_Cost; // evaluates the fitness of initial population.

Selection; // selects new population from parent based on fitness.

Mutation; // performs mutation operation on new population.

t = t + 1;

End while

Population_Cost; // evaluates the fitness of new population.

Our evolutionary algorithm work through a number of functions, that we show their flowcharts.

First: Initialize_Population Function creates the initial population of individuals based on the size of the population (PopSize), the following flowchart explains:

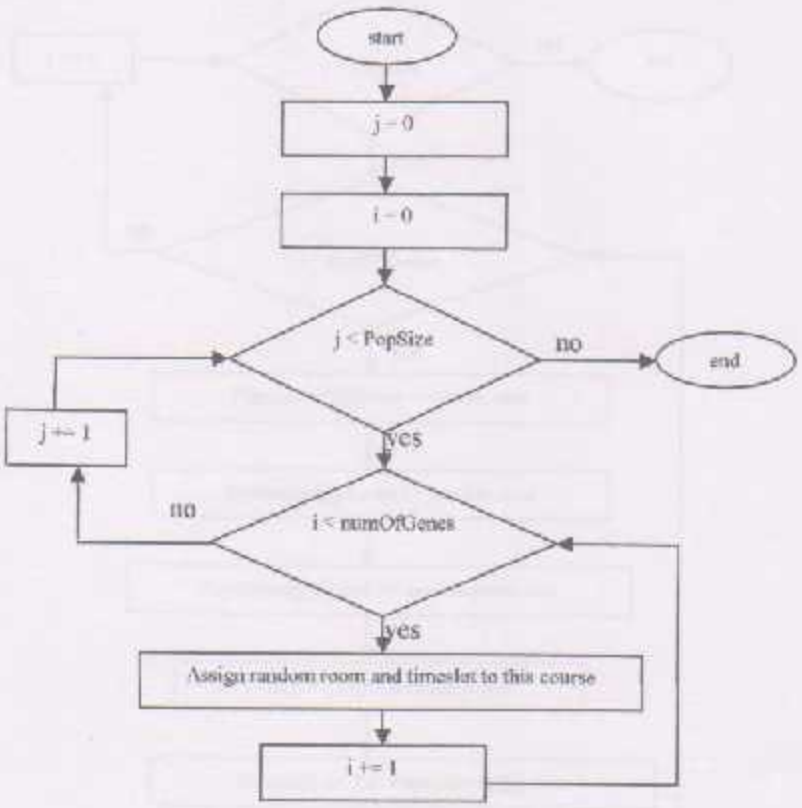


Figure (4.1): The flowchart of Initialize_Population Function.

Second: Evalute_Population function evaluates the fitness of the population as follow:
After we have built the population we evaluate the fitness of each individual in the population, the fitness depends on the cost of all genes in this individual, where the cost of each gene is the sum of costs of all constraints that the gene does not satisfy, and the fitness of each individual is calculated through the equation:

Fitness (G) = 1/cost (G) +1, where G: is an individual.

Figure (4.2) shows how we evaluated the Individual cost then the fitness of each Individual.

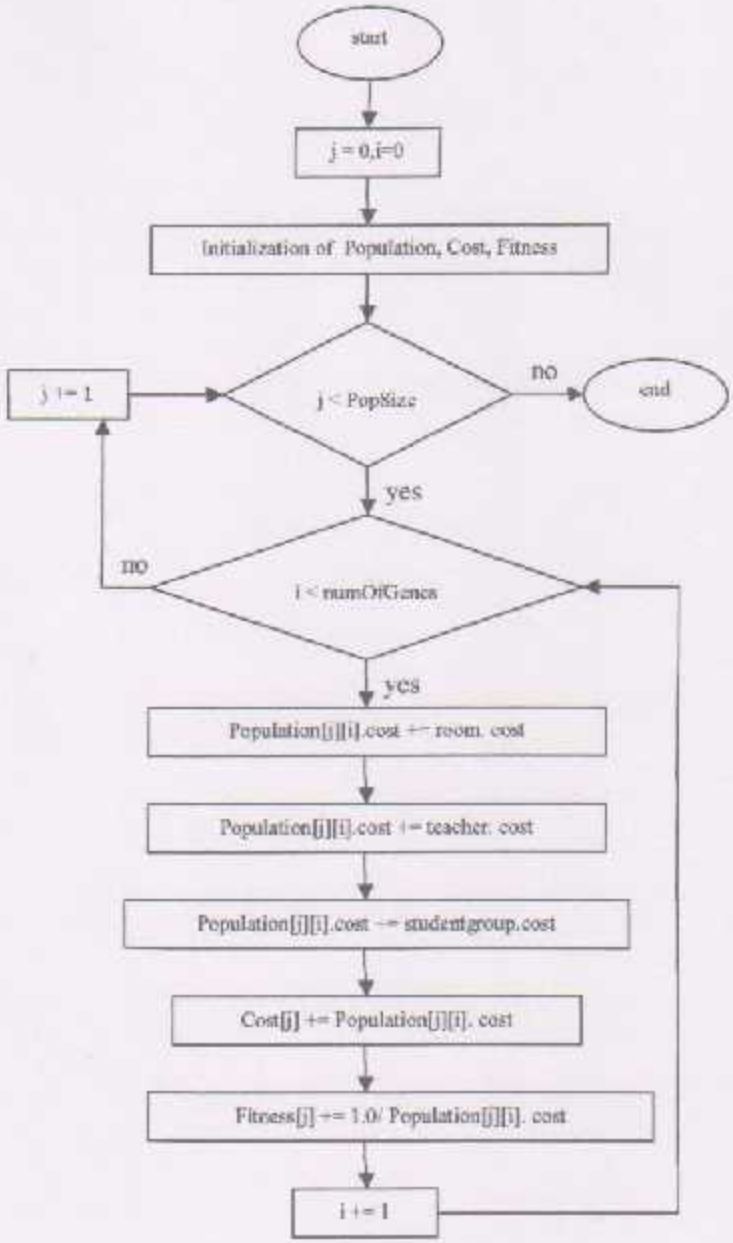
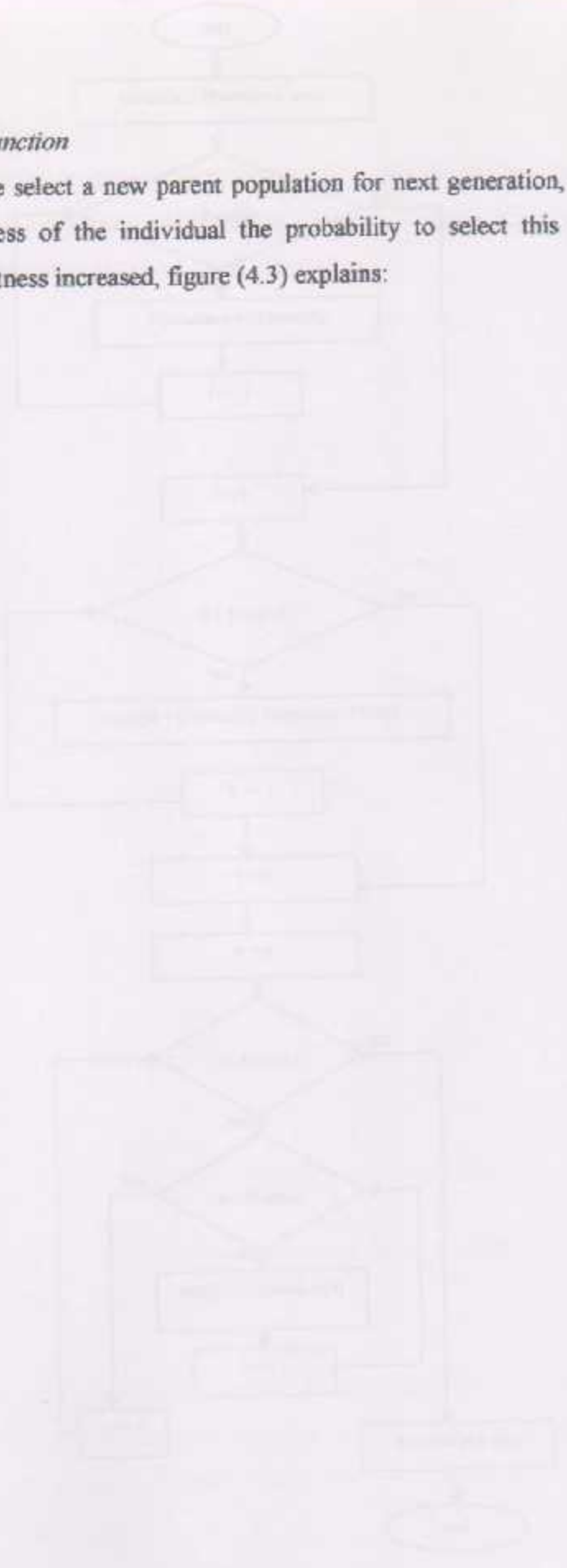


Figure (4.2): The flowchart of Evalute_Population Function.

Third: Selection Function

In this function we select a new parent population for next generation, the selection based on the fitness of the individual the probability to select this individual is increased as it is fitness increased, figure (4.3) explains:



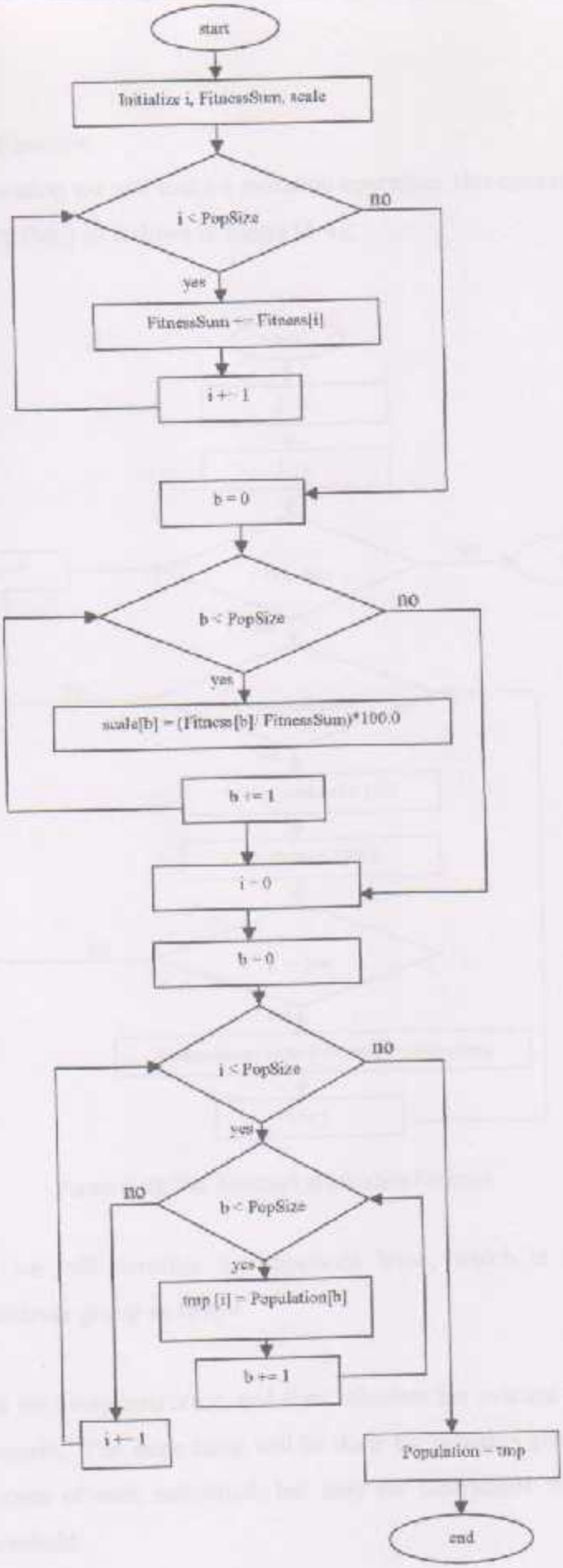


Figure (4.3): The flowchart of Selection Function.

Fourth: Mutation Function

After Selection operation we will make a mutation operation, this operation is done based on probability (Mu) as follows in figure (4.4);

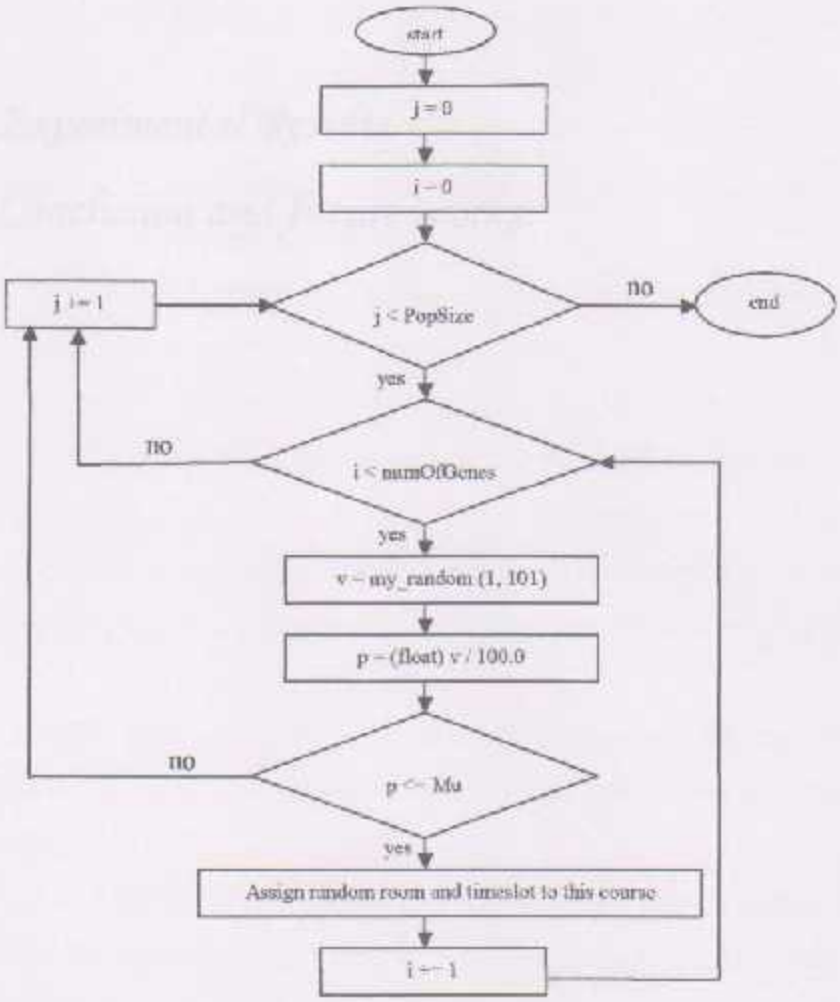


Figure (4.4): The flowchart of Mutation Function.

In our solution we will consider an important issue, which is the fairness for instructors and students group as follow:

Evaluate the cost for every instructor, and then calculate the average of these costs to determine the imparity. The same thing will be done for students groups. We will not consider the fairness of each individual, but only for individuals which have fitness greater than a threshold.

Chapter Five: Results and Conclusion

5.1 Experimental Results.

5.2 Conclusion and future works.

5.1 System Environment

Our application is implemented using standard C programming under Linux. Thus, we can easily modify the program to run in a parallel environment (Linux Cluster) in the future work. Our algorithm runs on Dual processor Intel (R) Xeon (TM) CPU 3 GH, cache 512 KB, total memory 2 GB workstation under Linux operating system.

In our experiments we use real data available on the database of Palestine Polytechnic University, and then the data is coded into numbers and stored in text files used as input to the program as follow.

Courses: this file contains complete information about all courses of the current semester. It contains number of student group that should register this course at the current semester, the course number, section number, teacher number for this course section, number of practical hours of this course, number of theoretical hours of this course, and the number of practical hours that the teacher must give.

Teacherstslots: this file contains the timeslots at a week for every teacher. In this file 0: means the teacher is available at this timeslot. 99: means the teacher is not available at this timeslot.

Roomstslots: this file contains the timeslots at a week for every classroom or lab. In this file 0: means the room is available at this timeslot. 99: means the room is not available at this timeslot.

Stdgroupstslots: this file contains the timeslots at a week for every student group. In this file 0: means the student group is available at this timeslot. 99: means the student group is not available at this timeslot.

The output from our experiments is coded into numbers and stored in text files used as output to the program as follow.

Teachersoutput: in this file we determined the timeslots for each teacher. Where 0: means free time. 1: means allocated timeslot.

Roomsoutput: in this file we determined the timeslots for each classroom or lab. Where 0: means free time. 1: means allocated timeslot.

Stdgroupsoutput: in this file we determined the timeslots for each student group. Where 0: means free time. 1: means allocated timeslot.

BestIndiv: this file represents the best solution, it contains the room number, lab number, theoretical time slot, and practical time slot for each course section.

5.2 Experimental Results

As the evolution time of our algorithm is greatly depending on the random function used the following figure (5.1) shows the nature of random function which we used in our experiments. It is clear that the function has Uniform distribution with mean 49.96362 and variance 832.741.

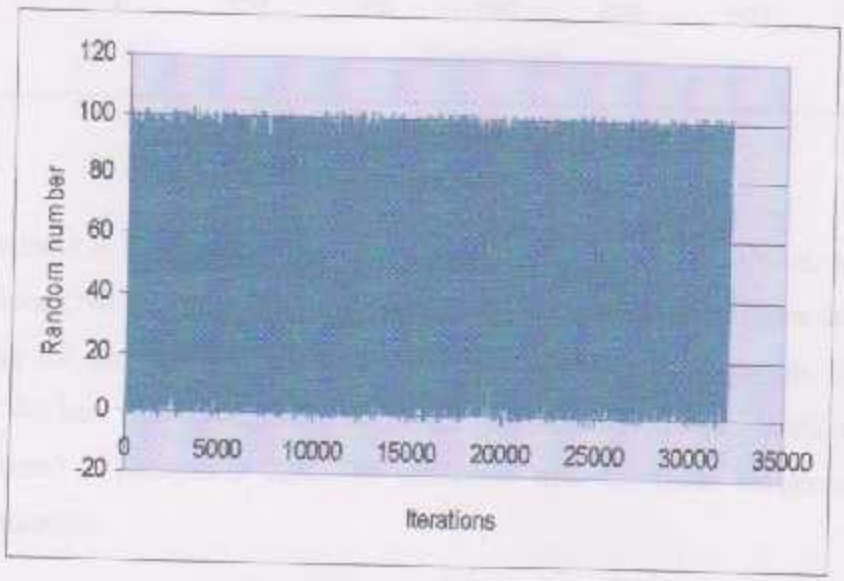


Figure (5.1): The curve of random function.

After doing many experiments these are some of the results. The following figure shows the relation between number of generations and the average fitness per generation. As we can see from figure (5.2) the evolution has been increased in a positive way; which means that the average fitness of the population increased while the repetition of processes.

Figure (5.2): The average fitness evolution.

This experiment is done with population size (PopSize = 30) individuals, and number of generations (NumGener = 500). We repeat every experiment 10 times then we take the average for the last six experiments to draw the following figures. After these iterations, the best individual has been found (the one which has minimum cost and highest fitness) in generation 228. The execution time for these iterations was 541 second (9 minute).

Figure (5.3) shows the relation between number of generations and the best fitness per generation. As we can see from this figure the average of best fitness evolution has been increased in a positive way.

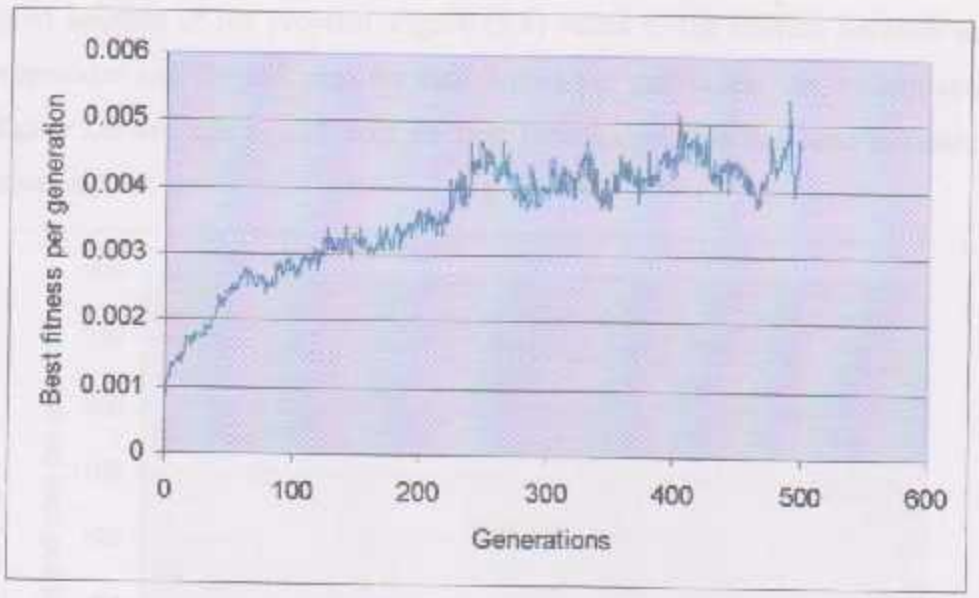


Figure (5.3): The best fitness evolution.

In the same way, figure (5.4) refers to the relation between number of generations and the hard cost for best fitness per generation. As we can see from this figure the average of hard cost for best fitness evolution has been decreased in a positive way

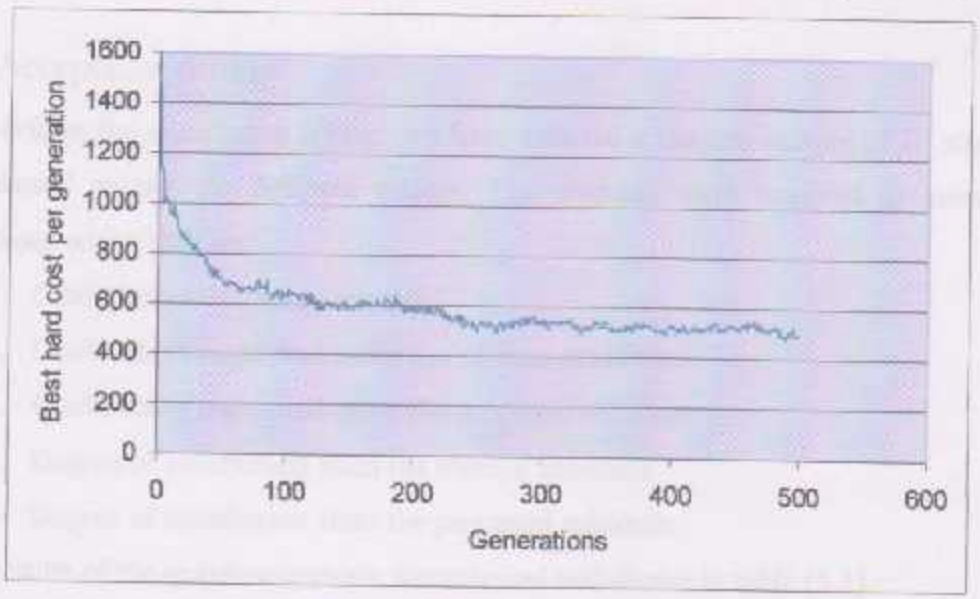


Figure (5.4): The hard cost evolution.

In relation to the soft cost as shown in figure (5.4) there is no evolution because bending the two objectives of the evolutionary algorithm require a definition of multi-objective optimization which in this case need more time and effort changing in the implementation. This can be considered more deeply in a future work; however our results are shown to be very good and feasible when comparing with the manually

prepared solution of the problem. Figure (5.5) refers to the relation between number of generations and the soft cost for best fitness per generation. As we can see from this figure the average of soft cost for best fitness evolution has been increased in a negative way.

Figure (5.5): The soft cost evolution

5.3 Acceptance testing

To perform the acceptance testing, we have selected a random sample of 20 students distributed among the different majors. The students were required to answer 5 questions which they are:

1. Credit hours wanted to register
2. Credit hours registered using the manual schedule
3. Credit hours registered using the proposed schedule
4. Degree of satisfaction from the manual schedule
5. Degree of satisfaction from the proposed schedule

The results of the questionnaire are summarized and shown in table (5.1).

		Sample	Students who had registration problems	Students who hadn't registration problems
Percentage		100%	35%	65%
Average credit hours wanted to be registered by the student		16.6	17.14	16.3
Average credit hours registered using the manual schedule		15.7	14.4	16.4
Average credit hours registered using the proposed schedule		16.5	16.7	16.4
Satisfaction from the manual schedule	Satisfied or very satisfied	60%	57%	62%
	Not satisfied	40%	43%	38%
Satisfaction from the proposed schedule	Satisfied or very satisfied	80%	100%	70%
	Not satisfied	20%	0%	30%

Table (5.1): Summary of results of the questionnaire.

5.4 Conclusion and future work

In this research we used evolutionary programming to solve one of the complex scheduling and difficult problem which is university courses scheduling. And we have succeed in achieving good and acceptable solutions for the problem, such solutions are shown to satisfy the hard constraints and getting a reduced cost of soft constraints.

We can summarize the achieved results in the following points:

1. A problem model is introduced suitable for the case of our university to get the most benefit from the project.
2. An evolutionary algorithm is proposed with special mutation operation which is proved to converge to good solutions.
3. A framework is implemented to construct a search space in order to apply the search methodology.
4. The proposed methodology is applied on a real data set from the collage of administrative sciences and informatics at the Palestine polytechnic university. With all its complexities and constraints.
5. Results show that our methodology can give better time schedule than those performed manually. In average, soft constrains were fulfilled up to 97% and hard constrains were satisfied up to 100%.
6. Acceptance testing shows that our work has solved registration problems for more than a third part of the students.
7. Also table (5.1) shows that the proposed schedule permits the student to register the credit hours they wanted to register. Furthermore, our proposed schedule has satisfied the student much more than the one put manually.

Future works

It's worthwhile that we mention here a number of future works that is desired to enhance the evolutionary algorithm itself and to achieve better solutions for the problem. Future works can be summarized in the following points:

- Using parallel implementation of the methodology to reduce the time, and get faster convergence.
- Modify the evolutionary algorithm to work as a multi-objective optimizer in order to reduce the soft cost.

- Trying other methods and evolutionary techniques such as PSO algorithm.
- Considering the problem in the other university's college. This means a greater and more complex search space.

References

[1] Ghaemi.S, Vakili.M, "Using a Genetic Algorithm Optimizer Tool to Solve University Timetable Scheduling", 2004.

[2] Pongcharoen.P, Promtet.W, Hicks.C, "Stochastic Optimisation Timetabling Tool for University Course Scheduling", International Journal of Production Economics, 2007.

[3] Opera.M, "Multi-Agent System for University Course Timetable Scheduling", The 1st International Conference on Virtual Learning, pp.231-237, 2006.

[4] Wasfy.A, A.Aloul.F, "Solving University Class Scheduling Problem Using Advances ILP Techniques", 2006.

[5] Petrovic.S, "Towards the Benchmarks for Scheduling Problems", 2006.

[6] Chitnis.S, Yennamani.M, Gupta.G, "NExSched: Solving Constraints Satisfaction problems with the Spreadsheet Paradigm", 2006.

[7] Gunawan.A, Ng.K, Poh.K, "Solving the Teacher Assignment-Course Scheduling Problem by A hybrid Algorithm", International Journal of Computer, *Information, and System Science, and Engineering, Volume1* number 2 ISSN *1307-2331, pp.136-141, 2007.*

[8] Yeung.V, "Declarative Configuration Applied to Course Scheduling", 2000.

[9] Dasgupta.P, Khazanchi.D, "Adaptive decision Support for Academic Course Scheduling Using Intelligent Software Agents", International Journal of Technology in Teaching and Learning, pp.63-78, 2005.

[10] Elmahamed.S, Fox.G, Coddington.P, "A comparison of Annealing Techniques for Academic Course Scheduling", 1998.

[11] Goldberg, D.E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley: Reading, MA, 1989

[12] Mitchell, M., Holland, J.H., and Forrest, S., "When will a genetic algorithm outperform hill climbing", In J. Cowan, G. Tesauro, and J. Alspector (Eds.), Advances in Neural Information Processing Systems. Morgan Kaufman, 1994.

[13] Murray, K., Muller, T 1, and Rudova H., "Modeling and Solution of a Complex University Course Timetabling Problem", http://www.unitime.org/, 2007.

[14] Kennedy, J., Eberhart, R., and Shi, Y., "Swarm intelligence",2001.

[15] en.wikipedia.org/wiki/np-hard.

Appendix

The following tables show the time table for every student group in the college of administrative sciences and informatics at Palestine polytechnic university obtained by our algorithm.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
استدراكي حاسوب	5054	3	4	312	4-3		4-3		4-3
لغة انجليزية 1	4003	3	10	221	12-11		12-11		12-11
الحاسوب واساسيات البرمجة	5055	3	6	222		12.30-11		12.30-11 5-2	
اللغة الانجليزية 2	4070	3	8	325	10-9		10-9		10-9
مقدمة في نظم المعلومات	4541	3	1	320		9.30-8		9.30-8	
مقدمة في الادارة	4503	3	1	325		11-9.30		11-9.30	
اللغة العربية	4001	3	4	310	9-8		9-8		9-8
مبادئ المحاسبة 1	4502	3	5	310	3-2		3-2		3-2
اللغة الانجليزية 1	4003	3	9	312	1-12		1-12		1-12
مقدمة في الاحصاء	5029	3	1	312		12.30-11		12.30-11	
الاقتصاد الجزئي	4247	3	2	218		2-12.30		2-12.30	
لغة انجليزية استدراكي	5052	3	3	221	12-11		12-11		12-11
الاقتصاد الجزئي	4247	3	1	223	5-4		5-4		5-4
مقدمة في الاحصاء	5029	3	2	110		2- 12.30		2- 12.30	

Table (A.1): Information systems second level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
تحليل النظم	5016	3	1	312		11-9.30		11-9.30	
مقدمة في نظم التشغيل	4259	3	1	110		12.30-11		12.30-11	
برمجة الحاسوب	4823	3	1		12-11		5-2		12-11
المحاسبة الادارية	5035	3	1	310		3.30-2		3.30-2	
السلوك التنظيمي	4265	3	1	310		9.30-8		9.30-8	
مبادئ التسويق	4543	3	1	310	11-10		11-10		11-10
اساليب البحث العلمي	4015	2	2	110		5- 3.30		5-3.30	
تركيب البيانات	4824	3	1	221	5-2		9-8		9-8
مبادئ تنظيم وعمارة الحاسوب	4257	3	2	320	1-12		1-12		1-12
مبادئ تنظيم وعمارة الحاسوب	4257	3	1	310		2 - 12.30		2- 12.30	
تركيب البيانات	4824	3	3	213	3-2		3-2		2-11

Table (A.2): Information systems fourth level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
الاقتصاد الكلي	4248	3	1	319	3-2		3-2		3-2
نظم المعلومات الادارية 1	4261	3	1	221	4-3		4-3		4-3
مقدمة في التجارة الالكترونية	4642	3	3	312		2-12.30		2-12.30	
ادارة تسويق	4270	3	1	218		-11 12.30		12.30-11	
الادارة الاستراتيجية واتخاذ القرارات	4268	3	1	110	5-4		5-4		5-4
نظم إدارة قواعد البيانات	4255	3	1	221	1-12		1-12		1-12

Table (A.3): Information systems sixth level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
مقدمة في ادارة المشاريع	4281	3	1	310	12-11		12-11		12-11
تاريخ فلسطين الحديث	4320	4	1	110	3-2		3-2		3-2

Table (A.4): Information systems eighth level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
ثقافة اسلامية	4002	3	5	110		9.30-8		9.30-8	
لغة انجليزية1	4003	3	11	310	4-3		4-3		4-3
لغة انجليزية 2	4070	3	11	221	3-2		3-2		3-2
مبادئ المحاسبة 2	4246	3	2	223		5- 3.30		5- 3.30	
مبادئ المحاسبة 2	4246	3	1	222		12.30-11		12.30-11	
اقتصاد جزئي	4247	3	3	310		12.30-11		12.30-11	
اقتصاد جزئي	4247	3	4	320		5 - 3.30		5- 3.30	
مقدمة في الادارة	4503	3	3	223		2 - 12.30		2 - 12.30	
مقدمة في الادارة	4503	3	2	223		11- 9.30		11-9.30	
مقدمة في الاحصاء	5029	3	3	223		3.30-2		3.30-2	
الحاسوب واساسيات البرمجة	5055	3	7	410	1-12		1-12		11-8

Table (A.5): Administrative management second level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
الثقافة الاسلامية	4002	3	7	110	4-3		4-3		4-3
المحاسبة الادارية	4264	3	1	222		12.30 - 2		2-12.30	
ادارة العمليات	4267	3	2	222	10- 9		10- 9		10- 9
ادارة العمليات	4267	3	1	110	9-8		9-8		9-8
مقدمة في نظم المعلومات	4541	3	2	319		9.30 - 11		11- 9.30	
مراسلات الاعمال	4544	3	1	312	12-11		12-11		12-11
مراسلات الاعمال	4544	3	2	223		8 – 9.30		9.30 – 8	
اساسيات البرمجة	4819	3	1	410	3-2	5-2	3-2		
اساسيات البرمجة	4819	3	2	410		9.30 - 11		11- 9.30 5-2	
انمنة المكاتب	4822	2	2	325	11-10	5-2			
انمنة المكاتب	4822	2	1	213	5-4			5-2	

Table (A.6): Administrative management fourth level.

اسم المسلق	رقم المسلق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
نظم المعلومات الادارية 1	4261	3	2	310	10-9		10-9		10-9
نظم المعلومات الادارية 2	4262	3	1	320	3-2		3-2		3-2
ادارة المشاريع الصغيرة	4269	3	1	222		11- 9.30		11- 9.30	
ادارة الشراء والمخازن	4567	3	2	312		3.30- 2		3.30- 2	
ادارة الشراء والمخازن	4567	3	1	312		5-3.30		5-3.30	
الادارة المحلية	4576	3	1	110	11-10		11-10		11-10
البرمجة المرئية لطلبة الادارة	4625	3	1	218		5- 3.30 2-11		5 - 3.30	
البرمجة المرئية لطلبة الادارة	4625	3	2	403		9.30-8		9.30- 8 2-11	

Table (A.7). Administrative management sixth level.

اسم المسلق	رقم المسلق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
قانون الاعمال	4271	3	1	325	5-4		5-4		5-4
اللغة العبرية	4308	3	3	222	4-3		4-3		4-3
اخلاقيات الاعمال والمسؤولية	4556	3	1	325	3-2		3-2		3-2
ادارة الاعمال العالمية	4558	3	1	221		9.30- 8		9.30- 8	
العلاقات العامة	4565	3	1	312	10-9		10-9		10-9
حلقة بحث	4644	1	1	312	11-10		11-10		11-10

Table (A.8): Administrative management eighth level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الأحد	الاثنين	الثلاثاء	الأربعاء	الخميس
اللغة العربية	4001	3	5	410	10-11		10-11		10-11
اللغة العربية	4001	3	6	110	12-11		12-11		12-11
اللغة الانجليزية 1	4003	3	13	319		3.30-2		3.30-2	
تفاضل وتكامل 1	4004	3	5	214		9.30-11		9.30-11	
تفاضل وتكامل 2	4005	3	15	218	8-9		8-9		8-9
تفاضل وتكامل 2	4005	3	13	312		8-9.30		8-9.30	
تفاضل وتكامل 2	4005	3	14	410		11-12.30		11-12.30	
اللغة الانجليزية 2	4070	3	10	320		12.30-2		12.30-2	
مبادئ المحاسبة 1	4502	3	1	110	12-1.30				12-1.30
مبادئ المحاسبة 1	4502	3	3	110	9-10		9-10		9-10
مبادئ المحاسبة 1	4502	3	2	223	8-9		8-9		8-9
مقدمة في الادارة	4503	3	4	218	10-9		10-9		10-9
مقدمة في الاحصاء	4507	3	3	223	12-1		12-1		12-1
برمجة الحاسوب	4823	3	3	218	11-12		11-12		8-11
برمجة الحاسوب	4823	3	4	325	3-4		3-4		2-5
برمجة الحاسوب	4823	3	2	213	2-5		10-11		10-11
مختبر نظم التشغيل	4892	1	1	PC2			2-5		
مختبر نظم التشغيل	4892	1	2	PC2		11-2			
مختبر نظم التشغيل	4892	1	3	PC4				2-5	
مختبر نظم التشغيل	4892	1	4	PC1		8-11			
الحاسوب واساسيات البرمجة	5055	3	8	213		12.30 - 2		12.30 - 2	

Table (A.9): Information technology second level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الأربعاء	الخميس
حلقة البحث	4279	3	1	218		2-3.30			
تاريخ فلسطين الحديث	4320	3	3	218	3-4		3-4		3-4
اللغة الفرنسية	4323	3	3	325		3.30-5		3.30-5	
هندسة برمجيات متقدمة	4550	3	1	218	11-10		11-10		11-10
البرمجة للانترنت	4615	3	1	222	11-10		11-10	11-8	
مقدمة في معالجة الصور	4617	3	1	325		2-3.30		2-3.30	

Table (A.12): Information technology eighth level.

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
اللغة الانجليزية 1	4003	3	14	223	10-9		10-9		10-9
علم النفس	4251	3	1	320	4-3		4-3		4-3
مقدمة في الادارة	4503	3	5	213	9-8		9-8		9-8
اساسيات التصميم 1	4731	2	1	مشغل		2-11		2-11	
الألوان ومزجها	4745	2	1	مشغل		5-2		5-2	
الرسم الحاسوبي	5146	0	1	312		11-8	3-2	11-8	
الرسم الحاسوبي	5146	0	2	320	2-11		12-11		2-11

Table (A.13): Graphics and multimedia second level.

اسم المسلك	رقم المسلك	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مبادئ الرسم الحاسوبي	4274	3	4	222		5-3.30		5-3.30	11-8
مبادئ المحاسبة 1	4502	3	4	218		11-9.30		11-9.30	
مقدمة في نظم المعلومات	4541	3	3	222		3.30-2		3.30-2	
البرمجة المرئية	4542	3	3	213	5-4	11-8	5-4		
معالجة التصميم 1	4732	2	1	ملتيميديا	11-8		11-8		
مبادئ الإبداع	4735	3	1	223	2-11	2-12.30		2-12.30	

Table (A.14): Graphics and multimedia fourth level.

اسم المسلك	رقم المسلك	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مبادئ التسويق	4543	3	3	223	4-3		4-3		4-3
البرمجة للوسائط المتعددة	4738	3	1	221		12.30-11		-11 12.30	11-8
تصميم الوسائط الرقمية	4741	3	1	312	9-8		9-8 5-2		
قانون الطباعة والنشر	4747	3	1	222	3-2		3-2		3-2

Table (A.15): Graphics and multimedia sixth level.

اسم المسلك	رقم المسلك	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
تصميم صفحات الانترنت	4643	2	1	319			9-8	11-8	9-8
تصميم الأفلام والرسوم المتحركة	4752	3	1	312			11-10	5-2	11-10
مواضيع خاصة	4755	3	1	213			4-3	2-11	4-3

Table (A.16): Graphics and multimedia eighth level.