إقـــرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

An optimization-Based Decision Support System for Higher Education Student Preferences-Based Scheduling (DSSPS)

أقر بأن ما اشتملت عليه هذه الرسالة إنما هي نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وإن هذه الرسالة ككل، أو أي جزء منها لم يقدم من قبل لنيل درجة أو لقب علمي أو بحث لدى أية مؤسسة تعليمية أو بحثية أخرى.

DECLARATION

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

Student's name: **Ahmad Fayez Muqbel Abu Libda**	اسم الطالب : أحمد فايز مقبل أبولبدة
Signature: [illegible]	التوقيع: أحمد فايز مقبل أبولبـــدة
Date: 27/07/2013	التاريخ: ٢٧/٧/ ٢٠١٣

بسم الله الرحمن الرحيم

Islamic University of Gaza

Deanery of Higher Studies

Faculty of Commerce

Department of Management

An Optimization-Based Decision Support System For Higher Education Student Preferences-Based Scheduling (DSSPS)

By

AHMAD F. ABU LIBDA

Thesis Advisors:

Prof. Salah Agha Prof. Yosuif Ashour

A thesis

Submitted in partial fulfillment of the requirements

for the degree of MBA

July_2013

عمادة الدراسات العليا

هاتف داخلي: 1150

الرقم ج س غ/35/ .... Ref

التاريخ 2013/07/24 .... Date

نتيجة الحكم على أطروحة ماجستير

بناءً على موافقة عمادة الدراسات العليا بالجامعة الإسلامية بغزة على تشكيل لجنة الحكم على أطروحة الباحث/ **أحمد فايز مقبل أبو لبدة** لنيل درجة الماجستير في **كلية *التجارة*/ قسم إدارة الأعمال** وموضوعها:

نظام دعم قرار أساسه تحقيق الأمثلية موجه للطالب الجامعي بهدف دعم عملية الجدولة الفصلية على أساس تفضيلات الطالب

An Optimization-Based Decision Support System For Higher Education Student Preferences-Based Scheduling (DSSPS)

وبعد المناقشة العلنية التي تمت اليوم الأربعاء 15 رمضان 1434 هـ، الموافق 2013/07/24م الساعة الحادية عشرة صباحاً بمبنى طيبة، اجتمعت لجنة الحكم على الأطروحة والمكونة من:

أ.د. يوسف حسين عاشور	مشرفاً ورئيساً	[illegible]
أ.د. صلاح رمضان الأغا	مشرفاً	[illegible]
أ.د. محمد توفيق حسين	مناقشاً داخلياً	[illegible]
د. سناء وفا الصايغ	مناقشاً خارجياً	[illegible]

وبعد المداولة أوصت اللجنة بمنح الباحث درجة الماجستير في **كلية *التجارة*/ قسم إدارة الأعمال.**

واللجنة إذ تمنحه هذه الدرجة فإنها توصيه بتقوى الله ولزوم طاعته وأن يسخر علمه في خدمة دينه ووطنه.

والله ولي التوفيق ،،،

عميد الدراسات العليا

[illegible]

أ.د. فؤاد علي العاجز

Abstract

With the rapid evolution of the computer and the increasing human dependence on it, new and innovative decision support systems are being designed continually to support and optimize decision making activities.

The objective of this study is to develop a decision support system based on zero-one goal programming and the analytic hierarchy process to aid the process of academic preferences-based scheduling in universities that adopt the credit hours system.

Usually, a high education student cares about making a satisfactory progress toward graduation; however, students usually have their own financial, timing or other personal issues regarding their study load.

The objective of this system is to provide the student with a schedule that optimizes achievement of his/her semester registration preferences, considering each one importance. These preferences are represented by the commonly considered ones, such as the desired number of credit hours, the desired empty days between final exam, the desired and the undesired group of courses, the proffered and the non-proffered lecturers and the desired empty days or periods throughout the week. Trying to reach these preferences, the system will also avoid all kinds of timing conflicts or breaching any of the commonly known registration regulations. Thus, the outcome of this system is a rapid, optimum and ready schedule.

The main component of this system is a computer software that serves as a linear programming models generator. This software – with the help of a backend database – generates different goal programming models for different cases, solve it and present the results in a readable way.

نظام دعم قرار يعتمد على البرمجة الخطية متعددة الأهداف و أداة التحليل الهرمي, يهدف لمساعدة الطالب الجامعي في عملية الجدولة الفصلية على أساس تفضيلاته الشخصية (ملخص).

مع التطور السريع الذي نشهده للحاسوب وزيادة اعتماد الإنسان له, تظهر باستمرار العديد من أنظمة دعم القرار الجديدة والمبتكرة و التي تساعد بشكل كبير عمليات اتخاذ القرار وتحقيق الأمثلية.

إن الهدف من هذا البحث هو تطوير نظام دعم قرار يعتمد على البرمجة الخطية متعددة الأهداف و أداة التحليل الهرمي, يهدف لمساعدة الطالب الجامعي الذي يتبع نظام الساعة في عملية الجدولة الفصلية على أساس تفضيلاته الشخصية.

إن من أكثر ما يهم الطالب الجامعي هو أن يحقق باستمرار تقدما مرضيا نحو التخرج, ولكن هذا الهدف عادة ما يتصادم و ظروفه الشخصية, والتي قد تكون مادية أو ذات علاقة بالوقت أو أي اعتبارات أخرى خاصة بالعبئ الدراسي الفصلي.

إن الهدف من هذا النظام هو تحقيق الأمثلية في الوصول إلى مجموعة من الأهداف المتمثلة في رغبات الطالب الشخصية المتعلقة بالعبئ الدراسي الفصلي آخذا بالاعتبار مدى أهمية كل منها. حيث تعد مجموعة المعايير المتضمنة في هذا النظام أكثر ما يهم الطلبة عادةً عند قيامهم بعملية تجهيز الجدول الفصلي, كعدد الساعات المعتمدة المرغوب و عدد الأيام الفارغة التي يرغب الطالب بأن تفصل بين الامتحانات النهائية و مجموعة المساقات المرغوبة والغير مرغوبة و قائمة المحاضرين المفضلين وغير المفضلين بالإضافة إلى الأيام و الفترات التي يرغب الطالب بتفريغها من المحاضرات خلال الأسبوع .

تم تطوير النظام ليتجنب أثناء عملية تحقيق الأمثلية جميع أنواع التعارضات اضافة إلى تجنب خرق أي من قوانين التسجيل المعروفة. وهكذا, فإن النظام سيزود الطالب بجدول دراسي سريع و أمثل و قابل للتسجيل في نفس الوقت.

إن لب النظام و أهم مكون فيه هو عبارة عن برنامج حاسوب, والذي يعمل كمولد للنماذج الخطية متعددة الأهداف. يقوم هذا البرنامج بمساعدة قاعدة بيانات مرتبطة بتوليد نماذج خطية مختلفة حسب المستخدم و حسب إدخالاته, ومن ثم يقوم بحلها وعرضها على صورة جدول دراسي واضح.

To my mother

ACKNOWLEDGMENTS

Thanks first to him from whom all life flows.

To my advisors, Prof. Salah Agha and Prof. Yousif Ashour for their supervision and guidance.

To thesis examiners, Prof. Mohammed Hussein and Dr. Sanaa' El Sayegh

To Mohammed El-Telbany - registration database specialist in the Islamic university of Gaza - for his help building the required query.

To anyone else that ever helped the author with his work.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ADO	ActiveX Data Objects
AGPA	Academic Grade Point Average
AHP	Analytic Hierarchy Process
ALGOL	Algorithmic Language
ANSI	American National Standards Institute
ASP	Active Server Pages
AVA	Academic Virtual Advisor
CBR	Case-Based Reasoning
CGP	Chebyshev Goal Programming
COM	Component Object Model
CR	Consistency Ratio
DAC	Data Access Components
DLL	Dynamic-Link Library
DSS	Decision Support System
DSSPS	Decision Support System for higher education student Preferences-based Scheduling
ES	Expert System
EXE	Executable
FAQ	Frequently Asked Questions
FAU	Florida Atlantic University
FROSH	An Expert System for Freshman Advisement
FROSH2	An Expert System for Freshman Advisement (Version 2)
GP	Goal Programming
GUI	Graphical User Interface

HTML	HyperText Markup Language
IPO	Input–Process–Output
IS	Information System
ISE	Industrial and Systems Engineering
LGP	Lexicographic Goal Programming
LP	Linear Programming
MCAL	Multi–Choice Aspiration Levels
MCDM	Multi Criteria Decision Making
MCGP	Multi–Criteria Goal Programming
MDAC	Microsoft Data Access Components
MILP	Mixed Integer Linear Programming
MODM	Multi–Objective Decision Making
OBDSS	Optimization–based Decision Support System
ODBC	Open Database Connectivity
OLE DB	Object Linking and Embedding, Database
OO	Object–Oriented
OODB	Object–Oriented Data base
OR/MS	Operations Research and Management Sciences
R&D	Research and Development
RAD	Rapid Application Development
RBKB	Rule–Based Knowledge Base
RDBMS	Relational Database Management Systems
RI	Random Consistency Index
RS	Rule Set
SAT	Scholastic Assessment Test
SMART	Simple Multi–Attribute Rating Technique

SQL	Structured Query Language
SWOT	Strengths, Weaknesses, Opportunities, and Threats
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
VB	Visual Basic
VBA	Visual Basic for Office Applications
WGP	Weighted Goal Programming

Glossary

Analytic hierarchy process: A structured technique for organizing and analyzing complex decisions.

Backend database: A database that is accessed by users indirectly through an external application.

Credit hour: Time-based reference for measuring educational attainment.

Database management system: A software system designed to allow the definition, creation, querying, update, and administration of databases.

Database: A collection of information designed to offer an organized mechanism for storing, managing and retrieving information.

Decision support system: A computer system designed to provide assistance in determining and evaluating alternative courses of action.

Expert system: A computer program that simulates the judgment and behavior of a human or an organization that has expert knowledge and experience in a particular field.

Goal programming: A branch of multi-objective optimization. It is an extension or generalization of linear programming to handle multiple, normally conflicting objective measures.

Linear programing: A specific case of mathematical optimization. It is a mathematical method for determining a way to achieve the best outcome in a given mathematical model for some list of requirements represented as linear relationships.

Multi Criteria Decision Making: A sub–discipline of operations research that explicitly considers multiple criteria in decision–making environments.

Operation Research: An analytical method of problem–solving and decision–making. It can also be defines as the Application of mathematical (quantitative) techniques to decision making.

Query language: A computer language used to make queries into databases and information systems.

The available classes: Classes represented by courses and their sections that are offered by the university for a certain student in a certain semester.

Zero–one goal programming: A special case of goal programming in which all the decision variables must have integer solution values of 0 or 1.

CHAPTER 1: General Introduction

1.1 Introduction

In a perfect world, a student's advisor at high education would only care about ensuring that the student is continually making a satisfactory progress toward graduation, so does the student.

However we do not live in a perfect world. Students usually have special circumstances. For example students may have financial issues as being unable to afford a certain number of credit hours. They may be unable to attend at certain times or at specific days for many reasons such as working part time while studying or being a parent or because of any other personal issues.

Sometimes their main priority becomes to empty a certain day from classes or to have final exams finish before a certain date, maybe because they need to travel, get a job or join an outside training course.

Besides, students may have a set of preferences about the classes they are going to attend regardless of their timing specifications such as the courses themselves because some students prefer to enroll in a certain combinations of courses – other than what is stated in their study plan prepared by the college – as they think they best fit with each other in the same semester, or because they would cause graduation delay if not taken. They may also specify groups of desired and undesired lecturers. Yet other goals can be considered a matter of concern form student's perspective such as the minimum number of days between final exams or the difficulty level of their study load.

Too many goals in mind with different importance while timing conflicts between classes and final exams continue to show up during the process of registration that may take the student too much time to overstep them. Even when the student manages to overstep these conflicts, he finds himself unsatisfied with the resulting schedule because it no longer or partially satisfies his/her goals.

Moreover, one can Easley notice the mess resulting from registration delay and instability by a significant portion of the students every semester, and how difficult it can be for the new students to decide what to register due to the confusion described before. This mess usually extends for more than a month each semester causing a number of academic and managerial problems. In this period Students make too many moves between classes, therefore in any class the students themselves and their number remain unstable, thus, it is impossible to follow them up or enforce the persevering term of not to miss more than 25 percent of any courses lectures applied in the Islamic university of Gaza. This will result in a decrease in their academic attainment because of the missed material that should have been covered by that time; this will in turn result in a decrease in the credibility of the education process itself of the university.

On the other hand, the credit hour system has too many advantages over the year system. For one thing, students enjoy more freedom when choosing courses and professors. It allows them to distribute their graduate requirements over the years of their study. There is also a substantial list of electives to choose from and one can delay taking certain courses till later. Furthermore, the credit hour system recognizes the principle of individual differences. Students who are unable to complete 18 hours a semester for one reason or another, can take 15 or 12 hours. In some cases, they can even take nine or six since some universities provide a part time educational system for students who cannot commit to the normal system. Thus Students have the opportunity to finish earlier, if they wish (i.e. three or three and a half years, instead of four) or finish later (i.e. five or six years).

In addition, while the yearly system allows students who are accepted in the same year to get to know each other well, the credit hour system allows one to get to know students from the previous and subsequent years as well. Furthermore, the credit hour system enables students study diversification, through the minors and double majors, this reflects positively on their employment opportunities as well as on their broadness of vision and it gives them more freedom in changing majors. On the other hand, if one fails a course, he will have to repeat this course alone if it is a compulsory course or take a substitute course if it is an elective.

Moreover, there are usually more than one exam per semester for each course in addition to the final exam, the daily quizzes and homework assignments which give the student more chances to do well. The multiple exams enable the students to get more feedback with respect to their performance, also simultaneously enabling the professor to assess the progress the students make far more accurately. Some educationalists have also argued that the credit hour system places more pressure on students and trains them (due to the shortness of the semester and frequency of exams compared to the yearly system) to work and think at a faster pace.

From the above, it's clear that the whole idea behind the credit hour system is to consider the students private issues and give them more freedom as for the various specifications of their semester study load, such as courses, lectures timing, professors or exams timing, however the question that remains is to how much the student can benefit from these privileges.

Most of the previous applications of the decision support systems in education field tended to concentrate mainly on the problem of scheduling times and places for the classes offered by a university in a certain semester, however, this DSS is directed greatly towards the student.

Other systems were developed to suggest an ideal schedule that guarantees a rapid and safe graduation to the students regardless of their issues, preferences or capabilities as discussed later in chapter two.

1.2 Research Objectives

The main objective of this work is to develop an optimization-based Decision Support System for higher education Student Preferences-based Scheduling (DSSPS). This system aims to maximize the student achievement of his/her semester registration goals considering each goal weight of importance while avoiding all kinds of timing conflicts represented in lectures timing or final exams timing. The optimization process will be within the framework of the general laws of the academic registration system at the university. The system is expected to provide the student with a ready, quick and optimal schedule from his point of view; this schedule will be in a form that is easily readable to the human eye. The system will also provide alternate solutions -as long as there are any- each time the user changes his input. When this system is placed in an online environment, it will be capable of counseling a numerous number of students simultaneously.

The research objectives can be summarized as follows:

A. Develop a system based on integer goal programming and the Analytic Hierarchy Process (AHP). The objective of this system is to provide the student with an optimal combination of classes from his/her point of view according to a set of criteria and weights that he/she should specify. The system will be capable of generating various multi-objective optimization models for various cases as for:
    a. Different student situations.
    b. Different specializations.
    c. Different goals and weights inserted by the user.
B. Evaluate the adoption potential of the system using a case study in the Islamic university of Gaza.

1.3 Population of the study

This study targets all universities that adopt the credit hours system. However, it will focus on universities in Gaza strip through the process of testing and data collection.

1.4 Research Importance

A successful implementation of this application will benefit the students as well as the university. As for students, this DSS will allow them to obtain an optimal combination of difficulty and credit hours, it will allow them to work part time while, at the same time, enabling him to graduate sooner. It will also enable them to enroll in their favorite courses with their favorite lecturers and be able to schedule final exams timetable the way they best feel comfortable with. All of these goals are roughly possible for a student to achieve while too many conflicts between lectures and final exams timing continue to show up. Consequently, student's registration process usually turns out to be a trial and error procedure which ends up with an unsatisfying schedule.

As for the university, a fast, computerized and scientific resolving of the previously stated confusing complex and repetitive matter which a vast number of students find themselves facing every semester, will be highly appreciated. Thus, in addition to the managerial and academic benefits of this system, its adoption as a student service is considered a competitive advantage and a point of strength to any university since it provides a sense of satisfaction and relief to the students by satisfying their goals while resolving timing conflicts.

Besides, the application of this system will significantly decrease the student online registering time and help him/her to select his/her classes as soon as possible. This will result in the highest academic attainment which in turn will result in an increased credibility and better reputation of the university.

Furthermore, this DSS will integrate with the academic advisor task since it does not only aim to satisfy personal schedule preferences but also include a criteria for the desired courses which can be filled with courses called for by the academic advisor.

From another point of view, this system can be used as an easy and effective information source for the criteria involved in it. If this system is imbedded in the university web site it can be prepared to store all choices made by the users. These data will be updated continuously and automatically so that it can be analyzed later to whatever end, for example in assessing lecturers, planning final exams timetable or lectures timetable.

Finally, this system will have the potential of being adopted by all universities that adopt the credit-hour system for registration which is the most commonly used system in universities everywhere nowadays.

1.5 Research Methodology

Research methodology explains the road map needed to reach the research goal. Figure 1.1 shows the methodology followed in this study.

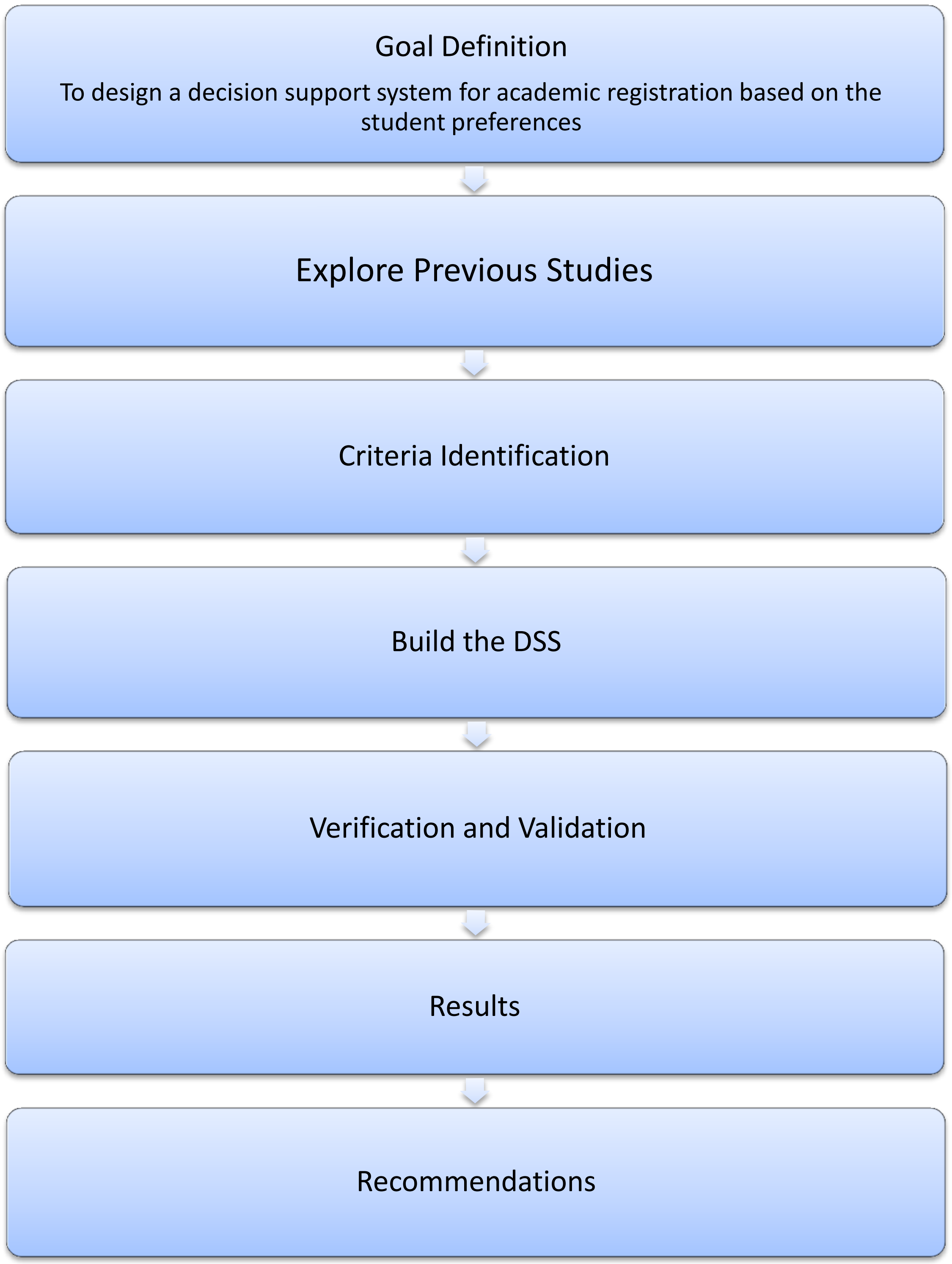

Goal Definition
To design a decision support system for academic registration based on the student preferences
Explore Previous Studies
Criteria Identification
Build the DSS
Verification and Validation
Results
Recommendations

Figure 1.1: Research Methodology

1.6 Goal Definition

The output of the DSSPS is represented in an easily readable schedule. This schedule consists of a combination of classes which –due to their characteristics– form an optimal solution for the intended user. Optimality here will be subject to the user input represented in a set of criteria associated with weights of importance. These criteria represent the various characteristics of a class such as the course itself, the course number of credit hours, lectures timing, subordinate lectures timing, professor or final exam timing. Based on the principle of goal programming, a satisfactory level of these criteria will be reached according to its associated weights of importance and to the extent that the available classes for the user in that semester allow. The system will also provide alternate solutions –as long as there are any– each time the user changes his/her input. The output schedule will be free of all kinds of timing conflicts and it will be within the framework of the general laws of the academic registration system at the university. This system can be placed later in an online environment to counsel a numerous number of students simultaneously.

The core of the DSSPS is represented by a computer software which works as a multi-objective model generator. The software will take input from the user about his/her criteria of interest, it will also incorporate the analytic hierarchy process in order to produce relative weights of importance among the inserted goals –as pairwise comparisons between the different criteria are supposed to be set by the user– then, based on the principle of goal programming and using the information acquired from the university database about the offered courses of the user in that semester, the system will generate a zero-one goal programming model that can be solved using one of the various linear programming engines. The resulting solution represented in zeros and ones will go back to the software to be translated to a readable schedule form. The resulting schedule can be called an optimal solution from the student perspective with respect to the specified criteria and weights; this result is mainly driven by the user insertion. Nevertheless, the system can lead to superb results when used under the supervision of an academic advisor.

1.7 A Brief Description of the System

1.7.1 Criteria Identification

As a start, a survey is used that contains a set of criteria which are – from the author's point of view – the most commonly considered by students, however, later it will be clear that other criteria can be easily involved in this DSS. These criteria are as follows:

A. The desired range of credit hours.
B. The desired number of specialization requirements, faculty requirements or university requirements courses.
C. Minimum empty days between final exams.
D. The furthest date of final exams.
E. The desired courses (with or without a preferred lecturer).
F. Number of empty days before a certain course final exam.
G. The undesired courses.
H. The undesired lecturers.
I. The desired empty days in schedule during the week.
J. The desired empty periods in schedule.

It should be noted that a student will neither necessarily consider all of these criteria nor that those that interest him have the same importance.

Through this study, students were asked to express their opinion about the previous criteria and whether there are other criteria that interest them.

On the other hand, the software itself can be used later to gather data about any other criteria that may be added to the model later.

1.7.2 Formulating the Integer Goal Programming Model and Constructing Its Generation Mechanism

This stage of the research involves a detailed description of the typical model that is generated by the software. It contains an explanation of the model variables, the model objective function and all types of constraints related to all of the criteria involved in the DSS in addition to the hard constraints of the model related to the general laws of the academic registration system. It also shows how the analytic hierarchy process was incorporated programmatically in the software and how its results are introduced to the model. Furthermore, this stage illustrates the algorithms of the generation approach followed by the software to construct the various types of constraints.

1.7.3 Building and Developing the Computer Program

This stage involves the process of the software developing. The code of the software is written in visual basic 6.0, the Structured Query Language (SQL) is also used to obtain information about the current user of the application.

From a programmer point of view, this stage includes two parts. The first part is the process of designing a flexible and efficient user interface while the second part includes the process of coding the software, however, practically, the software can be divided into four parts as follows:

A. Model generation part.
B. Analytic hierarchy process part.
C. Solving and translation Part.
D. Testing Part.

Other tools used to develop the DSS include:

A. Microsoft Access.
B. Lp_solve (a Mixed Integer Linear Programming (MILP) solver).

1.7.4 Information Feeding Mechanism

The software is built to be fed with information from a virtual database that is designed on Microsoft Access to integrate perfectly with the software, however, any university that wishes to adopt the DSS will have to develop a mechanism that is responsible of creating a database with the same design and the required information to be provided to the software from whatever database management system they are currently using. In this research an attempt is done in the Islamic university of Gaza with the help of the registration program specialists to reach this mechanism.

CHAPTER 2: LITERATURE REVIEW

2.1 Expert Systems.

In this section, briefly, previously created expert systems to handle the task of academic advising will be discussed.

2.1.1 Knowledge Engineering and Expert Systems.

An expert system can be thought of as a program with two components: a rule set (RS) and an inference engine. The RS consists of the information that the inference engine will process. Each piece of the RS typically contains two parts: the antecedent (ant) or condition and a conclusion (cul) coupled with a probability (prb). Because the rules form a set, these rules must be "syntactically different". The antecedents must be both sensitive and selective to insure that a conclusion will be "triggered" and that it is the correct conclusion. The inference engine is comprised of two pieces, a pattern-matcher and a conflict- resolution procedure. A basic approach would involve pattern-matching the antecedents of the rule with any new information found. If a pattern is found, the antecedents "triggered rule is added as new information. If it finds two rules triggered simultaneously, it uses a priori criteria to obtain a conflict resolution and get the best choice. After these steps, the process repeats with the updated data.

Expert system is meant to emulate human cognitive abilities. Because the results of a triggered rule can then become information later used to trigger another rule, the system produces a causal relationship to the data. One could even argue that it "learns" how to better deal with certain behaviors, similar to how a human learns which foods they like and which ones make them sick and how later on, they use this knowledge to avoid certain foods. Knowledge engineering refers to the process of creating an RS. Between acquiring the knowledge, testing it, and evaluating it, the knowledge engineer determines the rules and makes sure these new rules to not produce unanticipated problems with pre-existing rules. This is continued until there are no new rules [1].

2.1.2 An Expert System for Freshman Advisement (FROSH).

Advising freshmen is complicated by three factors: the heterogeneity of the skill levels of the freshman class, the fact that freshman advisors are usually expected to be able to advise students regardless of their prospective major, and the diversity within the core curriculum requirements. Frequently, this can lead to difficulties and delays during freshman registration as well as placing students into courses that do not meet the proper requirements for their prospective major. Expert systems are software programs that try to emulate the judgment of human expertise. FROSH2 is a rule-based expert system that guides the freshman advisor through planning a first semester's course schedule. It uses student data and information that appears in the advisor's handbook and university course schedule. FROSH2 then places the students in the classes appropriate for their placement scores, major, degree requirements, and academic preparedness, alleviating the aforementioned issues. Lastly, it gives the student a choice of one or more additional classes designed to meet general degree requirements compatible with the choice of major. The algorithms, rules, and design of FROSH2 are discussed as well as examples of class programs designed by it.

Siegfried, et al. developed FROSH to aid in advising freshmen at Saint Peter's College. The system was developed using the expert system development tool VP-expert and could be used as a consultant or as a training tool for freshman advisors. This was helpful because the academic diversity of the student body together with the twenty-nine majors available to incoming freshmen made the advisors' job difficult and error-prone.

FROSH helped the freshman advisor select a set of courses for the first-semester freshman. Initially, the user supplied basic data about the freshman including name, SAT scores, placement test scores and choice of major. FROSH first determined the maximum number of courses that a student should take, and then chose the appropriate composition and mathematics classes for the student. After determining the student's choice of major, it selected the appropriate beginning course(s) in that major for the student (or advised the student to wait until certain prerequisites were taken) and then helped the student choose additional courses until his or her program was complete.

FROSH had several shortcomings. It did not take course scheduling into account nor did it consider the possibility of course conflicts or closed and cancelled sections of courses. These deficiencies are being dealt with in subsequent versions [2].

2.1.3 Developing (FROSH2).

FROSH's biggest limitation is that it was built specifically to handle the advising needs of Saint Peter's College and any attempt to adapt it to another school's advising criteria required extensive modification. Additionally, the problems that are the most perplexing to advisors deal with unusual cases that FROSH did not and could not consider. These included cases where a student was planning to pursue two majors or a minor in addition to a major. So-called "double majors" and minors require the system to consider an additional set of requirements. But the most perplexing is the situation where a student needs to take two or three courses and the only available sections conflict. In such a case, the student usually postpones the course of lesser importance, but this "less important" course will vary depending on the students major.

The immediate goal in developing FROSH version 2 (also known as "FROSH2") was to explore the feasibility of a general framework for such an expert system, as well as handling the student's scheduling concerns. The latter is essential for such a system to be more than a toy or rudimentary teaching tool and the former is important if FROSH is to be made available to other colleges.

While two separate versions of FROSH2 were written, the basic algorithm was the same for both:

- Make sure that all necessary data were entered on the input frame. If not, display the appropriate error message.
- Determine the maximum number of credits/courses that the student must take.
- Determine the freshman courses to be taken by the student.
- Determine the major courses to be taken by the student.
- Allow the student or advisor to choose sections for these courses.
- Allow the student or advisor to choose other courses with which to complete the schedule.
- Make sure that there are no time conflicts and that the student is not taking multiple sections of the same course.
- Printing the complete schedule in Microsoft Excel.

VB6 was utilized for the rules and logic needed in advising, Microsoft Access was used to store the database of course sections, and Microsoft Excel was used to display the finished schedule as a spreadsheet [2].

2.1.4 A Web-Based Academic Advising System

At Florida Atlantic University. Personal interactions were found to cause inconsistencies in the advising process. Most of these inconsistencies involved answering recurring questions and the poor utilization of resources among the different advisors. Therefore, they set out to research and design a system that would provide stability in advising. However, most of the web-based advising systems they found were forums, PDF or HTML official documents available for download, useful links, or some amalgamation of the three. Through this research, several objectives were outlined for web-based advising:

- To minimize repetitive tasks currently performed by advisors.
- To encourage students to adopt a proactive attitude toward advising-related issues.
- To extend the availability of official advising-related information to remote students.
- To provide academic guidance in a consistent way
- To make advising-related information available in a single place, in electronic format.
- To maintain a (set of) HTML page(s) with the most frequently asked questions (FAQs).
- To develop a set of HTML forms and related ASP (Active Server Pages) scripts that allow a student to input the courses they have taken, press a button ("Advise Me") and get a list of courses to take next.

The resulting program was created using HTML, forms and ASP scripts. From its main page a user can access the requirements for their degree, a career guide, information pertaining to advising, and frequently asked questions. Most importantly, the user can access the form which allows him/her to input course information and personalized advice. The system supports three types of users: student users, faculty users, and administrative users, each with a different graphical user interface (GUI), appropriate rights and privileges, and set of actions. The students will use the system for advice, the faculty will update and manage information relevant to the FAQ page, and the administrative users will be responsible for the next courses to take module. All information that is regarded as classified or sensitive is password protected. Within the system there are 2 modules: FAQ and the next courses to take (hereafter known as 'Courses'). The FAQ module is a dynamically generated page that uses a backend database maintained by advisors. Questions are sorted across three categories: general, CS-specific, and CE-specific. Each question is input into the database with a unique key (identifier), category, question, answer, a date representing when it was last updated, and the name of who did the updating. The Courses module is designed to resemble the hard copy worksheets that are preexisting within the FAU CSE department. There are also three types of worksheets correlating to four-year students, transfer students, and second bachelor students. The backend database for the Courses subsystem consists of two tables, Course Info and Prerequisite. Course Info contains the course number, prefix, description, number of credit hours and type, which are the same three types previously mentioned for FAQ questions. Prerequisite has two input fields: one for the course to be taken and one for the course that is its prerequisite. After the Course subsystem retrieves input on courses available to be taken, it builds a directed graph based on the prerequisite information and does a topological sort. The designers of this system found several benefits to it. Not only did it increase the ability to access official information, but it also allowed answers to be found in a timely manner to most questions. Additionally, it decreased the amount of time advisors typically spent on recurring tasks along with a reduction in inconsistent advising [3].

2.1.5 A Prototype Student Advising Expert System Supported with an Object–Oriented Database

Using intelligent computer systems technology to support the academic advising process offers many advantages over the traditional student advising. The objective of this research is to develop a prototype student advising expert system that assists the students of Information Systems (IS) major in selecting their courses for each semester towards the academic degree. The system can also be used by academic advisors in their academic planning for students. The expert system is capable of advising students using prescriptive advising model and developmental advising model. The system is supported with an object–oriented database and provides a friendly graphical user interface. Academic advising cases tested using the system showed high matching (93%) between the automated advising provided by the expert system and the advising performed by human advisors. This proves that the developed prototype expert system is successful and promising [4].

The Object–Oriented Database (OODB)

An important objective in database design is to develop an efficient database structure so that data can be stored, accessed, and modified easily. Much of the work in creating an effective database is in the modeling. It is the application domain that determines how the database should be modeled in order to be successful. The nature of university subjects' and students' records (the domain of this research) reveals that the OO model is the most appropriate database modeling method. OO structure allows each course and each student to be constructed as a different object, and the database modeled as a collection of these objects. This structure gives more flexibility to each object to have whatever features (i.e. attributes or fields) required to identify it while maintaining the integrity of the whole system. The database of IS-Advisor consists of the main classes: Courses and Students. Figure 2.1 presents a portion of the object hierarchy of IS-Advisor which is the Kappa-PC's graphical representation of the OO database structure. Each study plan course in the database includes the following data: Title, ID, plan semester number (1 to 8), number of pre requisite courses, List of pre-requisite courses (if any), pre-requisite hours (Some courses have a specified number of hours as their pre-requisite), type of course (There are three types of courses: Compulsory courses, major elective courses, and university elective courses), keywords describing course contents (e.g. mathematics, programming, algorithm, management, marketing, etc.; these keywords are used to assist students in selecting courses based on their preferences as will be addressed later), course components (theory, lab, and/or tutorial), and course status (offered or not offered; note that fall -or odd- semester courses are offered in fall semester and spring -or even- semester courses are offered in spring semester). Each student object includes the following fields: ID, name, AGPA, passed compulsory courses, passed major elective courses, passed university elective courses, course grades semester-by-semester, earned credit hours, allowable courses, registered courses, course keyword preferences, and load preferences. Note that some data listed above are known and saved in the database (example: offered courses in a particular semester or AGPA of a student) and some data are inferred by the ES (example: lists of allowable and registered courses of a student). It is important to note that the proposed ES is intended to be used for course selection only, and based on courses selected by all students the timing of lectures will be determined manually by the timetabling committee in order to prevent the time conflict between courses. Thus the ES's recommended courses for students will be used as the input for the college timetabling committee. Therefore course timing is not a factor in the current version of the system and a component to automate the determination of lecture timings can be added to the system as a future work.

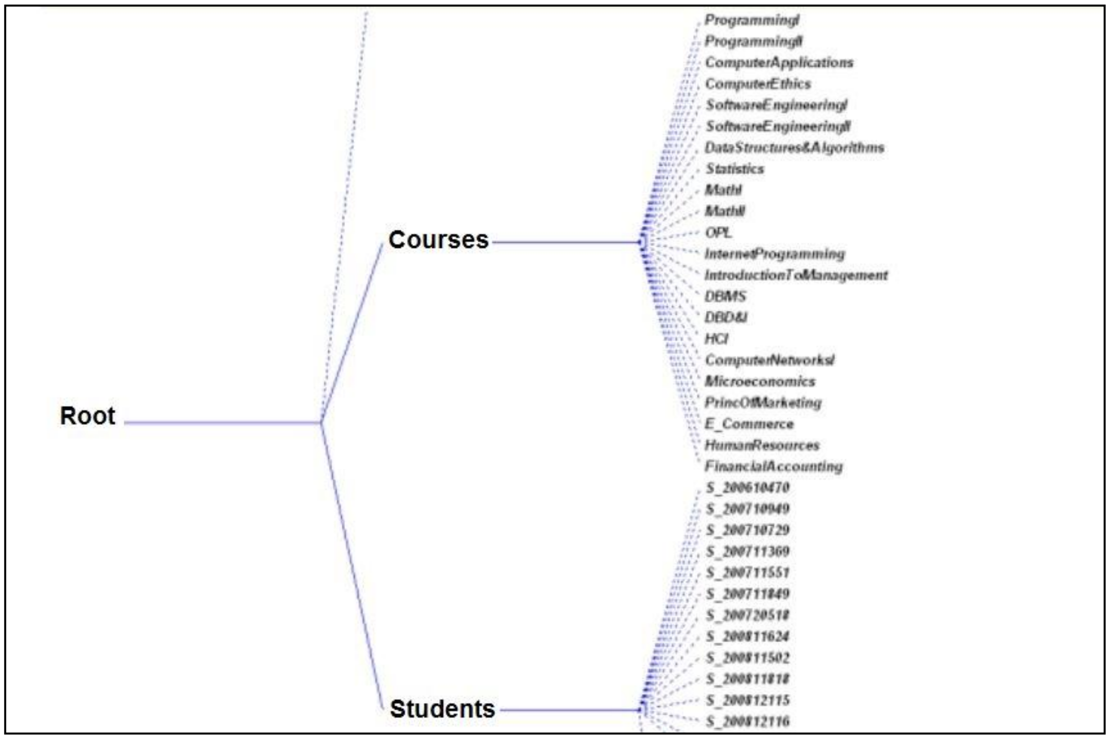


Figure 2.1: The Object hierarchy of the Object-Oriented database of IS-Advisor.

B. The Rule-Based Knowledge Base (RBKB)

The rules of the rule base can be classified into two categories: Academic rules and student-preference rules.

Academic rules are rules that are concerned with academic regulation like pre-requisites, the minimum and maximum number of courses that can be registered by a student (usually: minimum 3 courses and maximum 6 courses), etc.

As an example of this rule category, consider the following rules written in English:

Rule1:

If: The student passed Programming I AND Programming II is offered

Then: Add Programming II to the student's allowable courses list.

Rule2:

If: The student's passed hours are greater than or equal to 45 AND Computer Ethics is offered then: Add Computer Ethics to the student's allowable courses list.

Student-preference rules are If-Then rules related to preferences input by the student like preferred courses and preferred number of courses that the student is willing to register in a particular semester. As an example of this rule category, consider the following rule:

Rule3:

If: The student's course preference keyword is Management

Then: Mark all allowable courses having Management as a course keyword.

There are three main steps performed in the process of determining the recommended courses for a particular IS student. In Step 1 all courses that are offered and can be registered by the student are stored in a list called Allowable Courses. Step 2 performs the ranking process for the courses contained in Allowable Courses list. The courses are ranked in a descending order as following: (1) Courses that are pre-requisite for subsequent courses (have the highest priority), (2) Courses matching student preferences (in case preferences are given), (3) Courses officially in the current student's registration semester (fall or spring) according to the study plan, (4) Courses whose pre-requisites were passed in the previous semester (in order not to leave a long time gap between a course and its pre-requisite), and (5) Remaining 'equal' allowable courses (if any) are displayed to the user in order to rank them as preferred.

The list resulted from this step is called Ordered Allowable Courses. Step 3 is the filtering step that generates the ordered list of Recommended Courses based on the contents of the list Ordered Allowable Courses. This step follows one of the two advising models: Perspective advising (option 'One-Step Advising') or developmental advising (option 'Student's Preferences'). In 'One-Step Advising' option the list of Recommended Courses is generated as following: (a) Students with AGPA greater than or equal to 3.00 are given the courses ranked from 1 to 6 (from the Ordered Allowable Courses list). (b) Students with AGPA greater than 2.24 and less than 3.00 are given the courses ranked from 1 to 5. (c) Students with AGPA greater than or equal to 2.00 and less than 2.25 are given the courses ranked from 1 to 4. Note that if the remaining number of courses for a student towards graduation is less than the number of courses that can be suggested by the system, then the students is recommended to take the remaining courses only. In "Students' Preferences" option the student is asked to select the number of courses he/she is willing to register (3 to 6 courses) and course keyword preferences. Consequently the list Recommended Courses is prepared as explained in 'One-Step Advising' option above however here level 2 of ranking (courses matching student's preferences) is activated and the number of courses is equal to the number of courses selected by the student (if possible). In addition, more system messages are given here during the user-system interaction in order to guide the student to consider a 'more' suitable course selection [4].

2.1.6 Academic Virtual Advisor

In a perfect world, there would be one advisor for every student at every collegiate campus all across the globe. One advisor to ensure that each student not only made satisfactory progress towards graduation, but tailor made the student's academic schedules to best suit the student. But we do not live in a perfect world. We live in a world where the students vastly outnumber the academic advisors. With such a disproportionate number, time is of the essence. Advisors must find a way to determine each student's perfect schedule for typically hundreds of students.

Additionally, academic advising for an entire university often occurs in less than a month. Therefore, most students must find time in the approximately twenty business days to meet with their advisor, often being forced to meet with them before they are able to register for classes.

This limits each student to five to ten minutes to determine the next six to eight months of their academic career. And yet it often takes a student an entire afternoon to line up a possible schedule.

It is not uncommon for student records to be kept in a different building than the building within which a student will be advised in. Furthermore, it is often the student who must retrieve their own records and present them to their advisor. As the student does not hold a key to their own records, they must wait in line to have them located, trek across campus to their advisor where it is typical to wait in line again. This can become quite frustrating. But as technology becomes more prevalent and campuses become more wired, the registration lines of not too long ago seem archaic. And yet, if it is possible to register online and course information is already stored in a secure database, why is it that students still must wait in line to be advised?

The Academic Virtual Advisor (AVA) was designed with this in mind. Intended as a tool to alleviate the overcrowding of advisor offices, AVA can be used to supplement existing advisors. By using existing databases that contain student information and allowing advisors to create new databases stipulating available courses, course prerequisites and plan of studies for their departments (potentially in a less hectic portion of their schedule), AVA can be a surrogate advisor to most students.

AVA was also designed to leave the academic advisor in control of the advising process. While it can be used alone to assist a student, the advisor may choose to approve each potential schedule before a student registers. In the case of automatic approval, although not intended, AVA could be used to serve as a temporary advisor if an institution is currently lacking in human advisors. However, it is the web-based aspect of AVA that will assist human advisors and students the most. Since AVA is an online, database driven system, it is capable of supporting more than one student advising session at a time. Furthermore, each student has the ability to be advised where it is most convenient for them. By using the aforementioned existing student information databases, AVA eliminates the wait time students incur to retrieve their records and to be advised. Also, it eliminates the transit between record offices and advisor offices.

Using AVA, a student can be advised in as little or as much time as they would like, but is not forced to cancel their plans for an entire afternoon. In short, AVA is a customizable solution to the ever-increasing advisor to student ratio.

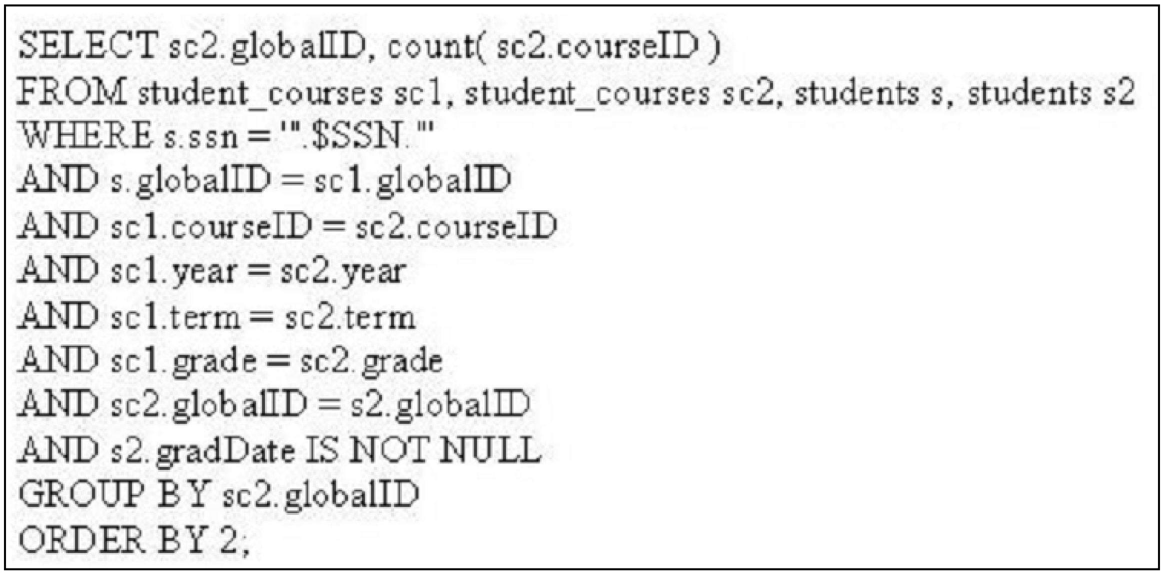
Schedule Creation

Case Based Reasoning

AVA uses Case Based Reasoning (CBR) to create one of the proposed schedules. As AVA was built with a database designed to simulate a real database of student records, fictitious data was created and inserted to provide cases. These fake records are full records of what classes the student took, when, and what grades they earned in that course. It also stores when they graduated and their GPA upon graduation, if appropriate.

After the student logs in, the system retrieves their completed courses, the grades they received, and when they completed the courses. It then matches, based on time and grade earned, those courses to another full record of a graduated student. Using the most accurate match, the student is then advised to take the courses that the matched case took to complete the curriculum.

These recommendations are checked to make sure the student has not already taken them. If so, they are no longer included in the possible schedule. This may leave the proposed schedule with fewer courses than needed to maintain a full load. In this case, the schedule is completed using the next courses in the chain made up by the plan of study.
The query for the CBR schedule is shown in Figure 2.2.



```
SELECT sc2.globalID, count( sc2.courseID )
FROM student_courses sc1, student_courses sc2, students s, students s2
WHERE s.ssn = '".$SSN."'
AND s.globalID = sc1.globalID
AND sc1.courseID = sc2.courseID
AND sc1.year = sc2.year
AND sc1.term = sc2.term
AND sc1.grade = sc2.grade
AND sc2.globalID = s2.globalID
AND s2.gradDate IS NOT NULL
GROUP BY sc2.globalID
ORDER BY 2;
```

Figure 2.2: **Case Based Query used in AVA**

Plan of Study Schedule

A plan of study can be thought of as a roadmap to graduation. In this situation, the students completed courses are considered nodes along that path. The completion of a node opens up other nodes as possible options (by completing prerequisites).

In the plan of study based schedule, the student's completed courses are compared to a listing of prerequisites. The courses that have their prerequisites met are listed in order of semester and year the plan of study advises them to be taken minus the courses that have already been taken.

The schedule is then formed by choosing the first five courses from this list. The query for the plan of study based schedule is the same query used to determine available courses during the information confirmation step [5].

2.2 Decision Support Systems (DSS)

In this section, briefly, previously created Decision Support systems to handle the task of academic advising will be discussed.

2.2.1 ADVISER

Designed at the University of Wisconsin in 1968, ADVISER was one of the first programs of its kind. ADVISER was programmed in ALGOL on 3000 cards with 22 methods.

It was developed not only to deal with the University of Wisconsin's course requirements, but to also handle the equivalencies generated by transfer credits. ADVISER is also not only for undergraduates, but also advises graduate level students, an aspect not often duplicated in other advising software. According to its algorithm, ADVISER first conducts an interview with the student to gather information on the student's completed courses and to conduct educated guesses concerning courses the student is unsure of having taken. It then calculates what the student's course load should be. However, it uses a great deal more math and statistics to determine a suggested course load for each student than other similar software. Adviser did have a study conducted on it. The study had eleven participants from various degree programs within the computer science department at the University of Wisconsin. Although the study had a small number of participants, the testers felt the diversity of the pool made up for it and validated their results. The study concluded that the interview process was far too long, that most participants were satisfied with the program, and that all would use it again if it were kept up–to–date. However, it was also found that not all enjoyed using it and that most of the subjects that were in graduate school did not enjoy using it and some were dissatisfied with the system.

Another important conclusion was the most students preferred a human advisor to a programmed one. Since advice is a subjective matter and cannot be measured holistically concerning its quality, it is impossible to determine if the advice of the computer was better or worse than the advice of a human advisor. Therefore, the only measure of quality is that of the student's perception, which is clearly slanted towards the human advisor, based on these results [6].

2.2.2 DSS for Academic Advising

This Decision Support System (DSS) was implemented to allow human advisers to focus on the more complicated problems rather than the more algorithmic course load selections. It takes into account the four types of academic courses: university requirements taken by all students (courses such as English, mathematics, and history), core requirements taken by students within a wide area (such as a college), major requirements, and electives. Unlike most of programs of its nature, this advising software was designed initially for business students rather than engineering, more specifically computer science.

A DSS is presented as an easier way to evaluate a student's progress towards graduation. It can also be a quick way to not just list courses that are required, but also those that the student is allowed to take in that the student has completed the necessary prerequisites. This calculation must be an error-free one for the system to have merit.

Since academic advising is typically a very structured process with the selection and sequencing of courses, the concept of a DSS can easily be applied. Additionally, an expert system (ES) can be used because the problem scope is quite narrow. Both methods include similar components of a knowledge base, an inference engine, and a user interface. Typically, an advising support system will use the plan of study for a major, taking into account the optimal scheduling to minimize semesters in school. This ignores course content and individual student issues. This particular Academic Advising software requires the student to input their own course information each time the system is used. Beyond this step, the program is designed quite similar to other advising systems. After the student has input his/her completed courses, the DSS produces a list of eligible courses and completed courses using binary categories within the database.

Also used to choose eligible courses and more specifically, their order, are three hierarchical rules. The first rule is the 'Deepest Layer Rule' which chooses courses on the deepest level of prerequisites first and then choose courses based on the descending order of their layers. Next is the 'Maximum Dependency Rule', which sorts courses within each layer by the number of prerequisites they will complete. Lastly, the 'Course Number Rule', which chooses courses based on the ascending order by their course number. It is concluded that this DSS will work, but is not the best option. Instead, a database management system is recommended. Since the database is the largest part of the DSS it becomes difficult to separate it from the inference engine, a DSS generator would create a more flexible user experience as well as provide an easier method of maintaining degree requirements [7].

2.2.3 Virtual Academic Advisory "A solution using Integer Linear Optimization".

Describes the research done to create a Web Decision Support System that suggests alternative feasible course schedules that a student can choose for the upcoming semester. The research has 3 main components: Coding of the computer program, formulation of an integer program, and the gathering of meaningful data from students and department of industrial and systems engineering (ISE) in the university of Florida.

In a typical session of the application, a student (user) will log in using his Gator link and Password. The program will obtain from a database relevant information about the current user, including name, major, courses already taken, unmet requirements, and prerequisites for the unmet requirements. The user will then select the courses he wishes to be included in the suggestions, the degree of desirability of each time period (e.g. 7:25 AM is very undesirable), and a range of number of credit hours that he wants to take on the upcoming semester (e.g. between 12 and 14 credit hours).

The application will then create multiple feasible course schedules and display them so that that the user can choose the one he finds the most attractive for the upcoming semester.

The application can be considered in part as an “automated academic adviser” because it will provide students with multiple options that will be laid out in an easy–to–read manner.

A human academic adviser will still be required because the application does not provide enrollment capabilities or other important functions.

The application will be capable of gathering information from an individual student (user) to determine relevant parameters in order to suggest a meaningful combination of classes.

Scope and Limitations

To develop the application, the majority of the effort was allocated to a proper formulation and partial coding. An attractive and embellished user interface is not yet a characteristic of the software. More importantly, the program does not yet output alternative course schedules easily readable to the human eye. Instead, the program currently specifies which periods will be filled with which class. The final step of the Web DSS project will be to use this information to create schedule tables to lay out the optimal combination of courses.

As of now, the code of the application does not contain all of the information gathered throughout this research project. Instead, it uses synthetic data. As previously mentioned, this research focused on gathering the necessary information to enhance a class project developed in the Web DSS graduate course of spring 2010. Including this information in the code would require a considerable amount of time. Nevertheless, it is not expected it would alter the functionality of the system when inserted.

In addition, due to time constraints and simplicity, the application was developed using information, such as classes, about the ISE department only. Nevertheless, the program is capable of easily incorporating other departments and majors, along with the courses and sections that they entail. To do this, the database supporting the application would need to include this information.

For security reasons, all of the data used in the application is synthetic. However, the data obtained from the set of surveys is real. The names of the surveyed are not supplied because it is irrelevant and could constitute a privacy issue.

Currently, the application is only useful for university of Florida students due to the fact that some sections, such as course schedules, have meeting times that are hard coded in the system. Integrating the program to another University would require certain modifications to the code. Also, the course planner does not support the suggestion of courses to be taken at a different school, which is a common thing to do among students during the summer semesters [8].

2.3 Comments.

All of the studies viewed before concentrate mainly on the problem of the traditional counseling, as how to present the student with the best course plan that guarantees the shortest route towards graduation through the completion of all of his major requirements.

These systems are trying to imitate the role of the academic advisor, so, they focused only on what courses should be taken next depending on what the student has completed so far and what offered courses are now available for him. They even hardly handled the issue of the conflicted sections, but for sure have not taken the student preferences into account except for the study of Andres Scharifker (2010) which will be discussed deeply later.

Because of the above, due to the prerequisites issue and the fact that it is not logical to suggest courses that have been taken already, most of the systems discussed before are rule–based expert systems that is meant to emulate human cognitive abilities depending on the information they get about the dealt case. Because of that, these systems can be applied only in universities in which they were first developed. This is because there are some rules and information related to these universities that are hardly coded into these systems, consequently, these systems will require extensive modification to adapt to another universities.

On the other hand suggesting courses for the upcoming semesters is a bit vague, because there is no knowing whether a student is going to pass his suggested courses or not and whether a certain courses will be offered by the collage on a certain semesters. That is why this issue was handled in the study of “Academic Virtual Advisor” for Kathryn Nobles using case based reasoning where the system matches – based on time and grade earned – a student records to another full record of a graduated student. And then advise the student to take the courses that the matched case took to complete the curriculum.

However, the decision support system of this study is less interested in enhancing a student progress towards graduation than it is in satisfying his personal preferences regarding the current semester. All possible aspects that may concern a student regarding the classes he is going to attend are addressed. These aspects include the desired number of credit hours, the number of each type of courses, the minimum period between final exams, the desired last date of final exams, the desired and the undesired courses, the desired and the undesired lecturers and the desired empty days or periods throughout the study schedule.

The main point that features the approach of this tool over other DSS's is the use of goal programming and AHP being a model-based DSS. Goal programming can be thought of as an extension or generalization of linear programming. Thus it depends mainly on the mathematical representation of a problem in the form of an objective function that is subject to a set of constraints. A mathematical approach is a good reason as why the output will be considered a robust and undeniable solution. On the other hand, the use of the analytic hierarchy process will ensure the maximum prioritization accuracy among the various goals by maintaining the lowest level of inconsistency through pairwise comparisons.

As integer programming was used as a base to construct this DSS and due to the efficient way in which the model and its generation mechanism was created, this system has the ability to be adapted in any university directly, or maybe with a very slight modifications. Besides, it is capable of including any other criteria easily as they may show up later.

2.3.1 Virtual Academic Advisory "A solution using Integer Linear Optimization", Andres Scharifker (2010) – An Extensive Analysis.

1. **Formulation of the integer program:**

- The objective function.

$$\text{MIN} \quad \sum_{i=1}^{m}\sum_{k=1}^{s}\sum_{j=1}^{n} A_{ijk}\left(\mathrm{w} * c_j * x_{ijk} + (1-\mathrm{w}) * u_i * x_{ijk}\right) \tag{2.1}$$

- Constraint #1: Ensures that 2 different courses are not assigned to the same period.

$$\sum_{i=1}^{m}\sum_{k=1}^{s} A_{ijk} * x_{ijk} \leq 1 \quad \forall j \tag{2.2}$$

- Constraint #2: Ensures that the difficulty level of a specific suggested schedule is within constants DL and DU.

$$DL \leq \sum_{i=1}^{m}\sum_{k=1}^{s}\left(\frac{\mathrm{d}_i}{\mathrm{p}_i} * \sum_{\mathrm{j}=1}^{\mathrm{n}} A_{ijk} * x_{ijk}\right) \leq DU \tag{2.3}$$

- Constraint #3: Ensures that the number of credits suggested is bounded

$$L \leq \sum_{i=1}^{m} \sum_{k=1}^{s} \left(\mathrm{r}_i * \frac{\sum_{\mathrm{j}=1}^{\mathrm{n}} A_{ijk} * x_{ijk}}{\mathrm{p}_i} \right) \leq U \tag{2.4}$$

- Constraint #4: Ensures that a certain course is not only partially assigned to a schedule. In other words, it guarantees that if a course will be suggested, it will appear in all of its corresponding periods.

$$\sum_{j=1}^{n} A_{ijk} * x_{ijk} = p_i * B_{ik} * y_{ik} \quad \forall\, i, \forall\, k \tag{2.5}$$

- Constraint #5: Ensures that if a section of a course is suggested, no other section of that course will also be suggested in the same schedule.

$$\sum_{k=1}^{s} B_{ik} * y_{ik} \leq 1 \quad \forall\, i \tag{2.6}$$

Variables and Constants of the Integer Program

xijk = 1 If section k of class i is assigned to period j, 0 otherwise – Decision Variable

yik = 1 If section k of class i is assigned, 0 otherwise – Decision Variable

i : Courses considered for suggestion. Each i represents one course –> i ∈ [1, 44].

m: Number of Courses considered for suggestion (i.e. i : 1, …. ,m)

j : Periods in a schedule. Each j represents one period–> j $\in [1, 75]$ (See next page)

n: Number of periods considered for suggestion (i.e. j : $1\ldots$, n)

k : Sections. Each k represents one section.

s : Number of Sections considered for suggestion that belong to a specific course

w : Weight Factor – Constant

cj : Cost of assigning a class to period j – Derived Constant

ui : Cost of assigning class i – Derived Constant

ri : Number of credits that class i entails – Constant

di : Difficulty associated with Class i – Constant

pi : Number of meeting periods required by class i – Constant

DL: Constant denoting the minimum level of difficulty desired for the schedules – Constant

DU: Constant denoting the maximum level of difficulty desired for the schedules – Constant

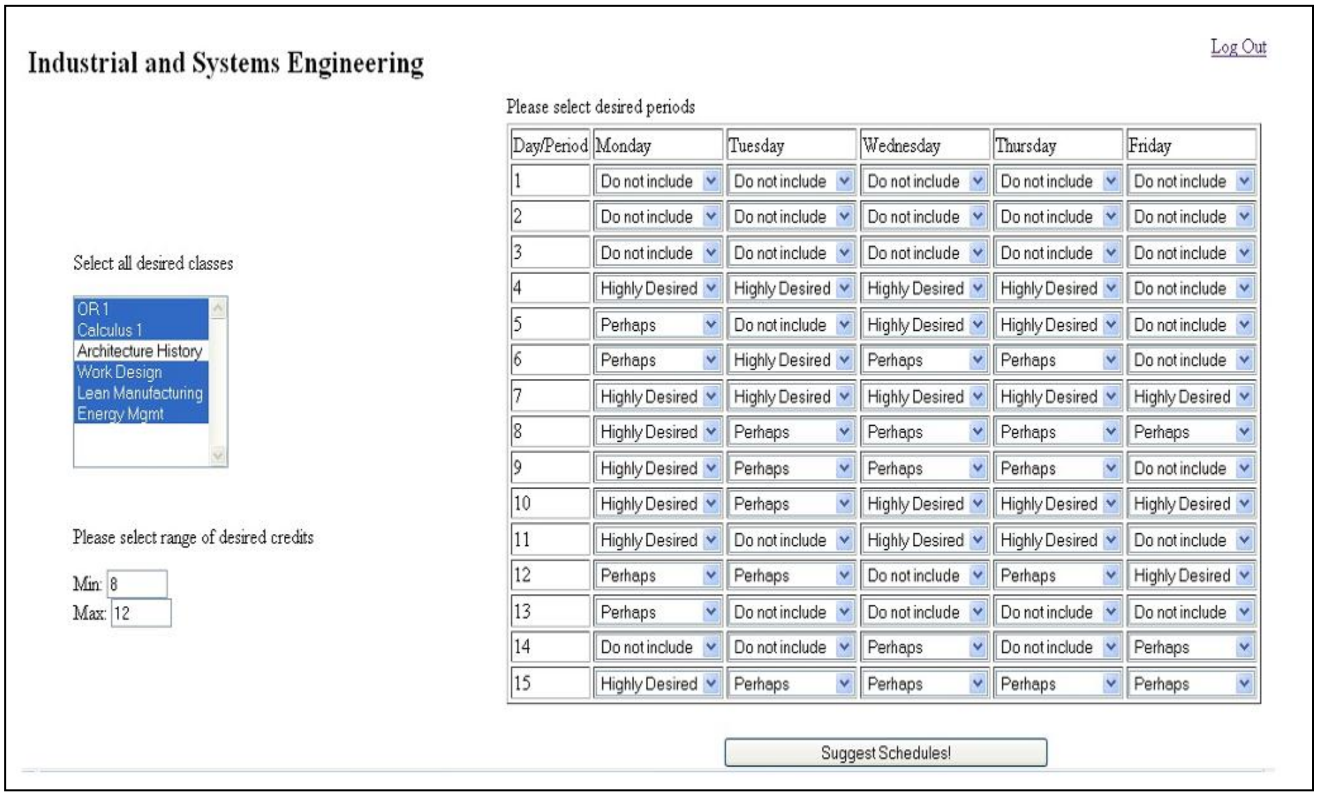
L: Credits Lower Bound, and it is a constant that denotes the minimum number of credits considered for suggestions – Constant

U: Stands for Credits Upper Bound – Constant

Aijk: Binary constant existing for each combination of cours (i) period(j) and section(k)

Bik: Binary constant existing for each combination of course (i) and section (k)

The current user selected the periods for which he has a preference, and also specified on the range of number of credits he is willing to take as shown in figure 2.3. He also selected which courses he wants included in the suggestions. If he does not select any, the program will assume he wants all the courses included in the suggestions [8].



Industrial and Systems Engineering

Log Out

Select all desired classes

- OR 1
- Calculus 1
- Architecture History
- Work Design
- Lean Manufacturing
- Energy Mgmt

Please select range of desired credits

Min: 8

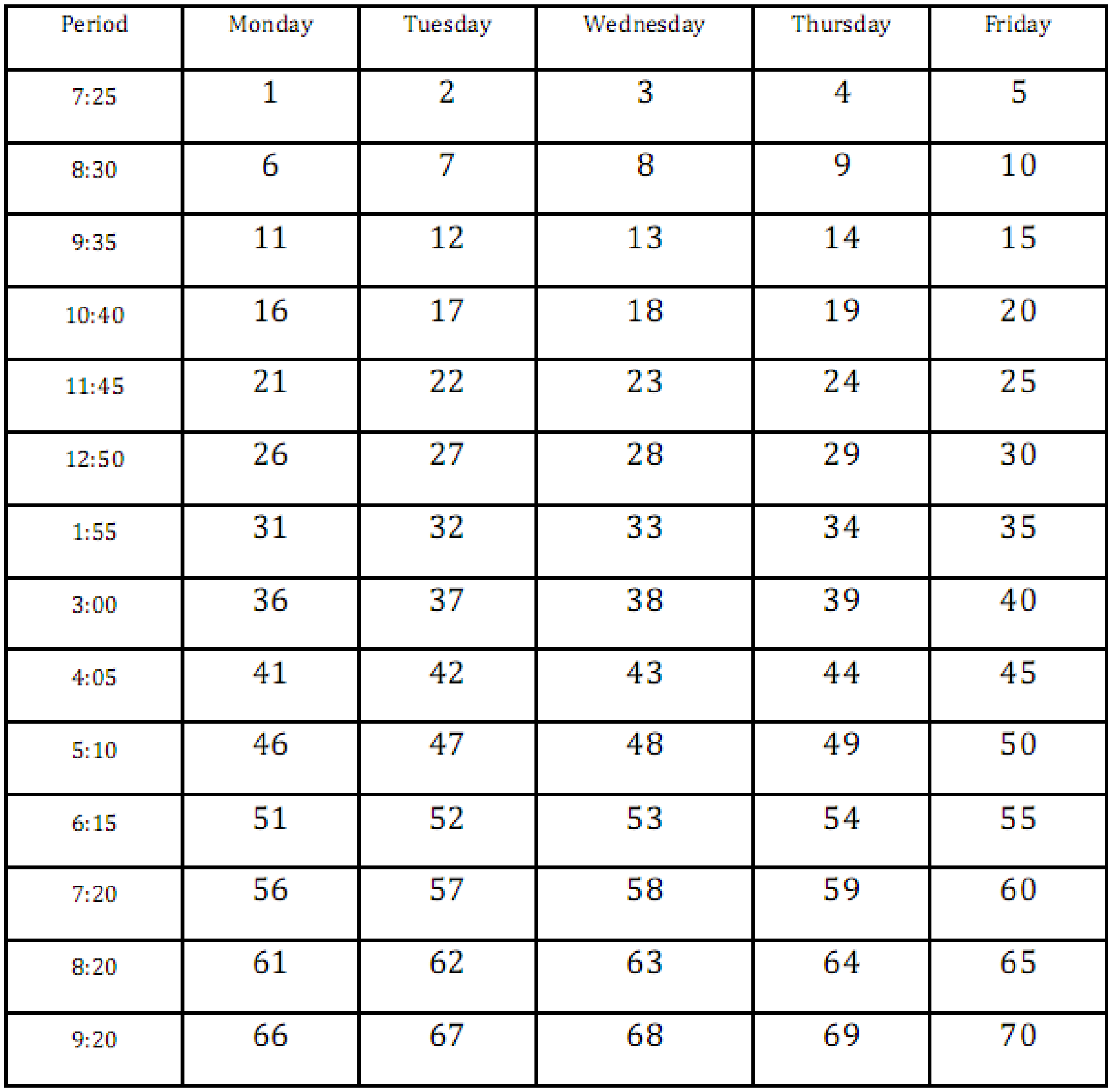
Max: 12

Please select desired periods

Day/Period	Monday	Tuesday	Wednesday	Thursday	Friday
1	Do not include	Do not include	Do not include	Do not include	Do not include
2	Do not include	Do not include	Do not include	Do not include	Do not include
3	Do not include	Do not include	Do not include	Do not include	Do not include
4	Highly Desired	Highly Desired	Highly Desired	Highly Desired	Do not include
5	Perhaps	Do not include	Highly Desired	Highly Desired	Do not include
6	Perhaps	Highly Desired	Perhaps	Perhaps	Do not include
7	Highly Desired	Highly Desired	Highly Desired	Highly Desired	Highly Desired
8	Highly Desired	Perhaps	Perhaps	Perhaps	Perhaps
9	Highly Desired	Perhaps	Perhaps	Perhaps	Do not include
10	Highly Desired	Perhaps	Highly Desired	Highly Desired	Highly Desired
11	Highly Desired	Do not include	Highly Desired	Highly Desired	Do not include
12	Perhaps	Perhaps	Do not include	Perhaps	Highly Desired
13	Perhaps	Do not include	Do not include	Do not include	Do not include
14	Do not include	Do not include	Perhaps	Do not include	Perhaps
15	Highly Desired	Perhaps	Perhaps	Perhaps	Perhaps

Figure 2.3: Preferences Page of the Virtual Academic Advisory developed in the study of Andres Scharifker (2010).

Figure 2.4 shows the way by which periods throughout the week were defined.



Period	Monday	Tuesday	Wednesday	Thursday	Friday
7:25	1	2	3	4	5
8:30	6	7	8	9	10
9:35	11	12	13	14	15
10:40	16	17	18	19	20
11:45	21	22	23	24	25
12:50	26	27	28	29	30
1:55	31	32	33	34	35
3:00	36	37	38	39	40
4:05	41	42	43	44	45
5:10	46	47	48	49	50
6:15	51	52	53	54	55
7:20	56	57	58	59	60
8:20	61	62	63	64	65
9:20	66	67	68	69	70

Figure 2.4: Representation of periods throughout the week defined in the study of Andres Scharifker (2010)

Comments:

- Variables of the model were defined in such way that makes it very complex to construct the various relations of the model.

  Variables were defined to express the possibility of the existence of the combination of a section (k) of a class (i) to be in a period (j) for all 75 periods and all sections of all classes. The same number of variables were also defined for the possibility of assigning a certain combination to the schedule What results in a very large number of redundant variable, although timing in which the various sections meet is already defined and fixed by the university..

- The system addressed only four types of student preferences which are:

  A. Desired courses.

  B. Undesired periods throughout the week.

  C. Desired range of credit hours.

  D. Desired range of the difficulty of the study load.

- Only undesired periods and undesired courses are penalized in the objective function while the range if the desired difficulty and the range of the desired number of credit hours are expressed as hard constraints. This may result in infeasible solutions as they may conflict with each other's or with the available courses and sections.
- The cost of each time slot is already defined using the information obtained from a survey. These costs is hard coded in the application, then a variable "w" is used in the objective function to determine the relative importance of both the undesired periods costs and the undesired courses costs, although these costs may differs from a student to another.

- As for the desired courses. The application present a list with all available courses so that the user will select his desired group of courses, thus, the application will consider the reminder of the courses as undesired and place a penalty for them as they are assigned.

  However, to specify two desired and undesired courses groups gives the student more control as in this way courses will be classified into three groups which are desired courses, undesired courses and courses that do not matter.

  Moreover:

  A. It will be easier for the student to specify two small groups of desired and undesired courses leaving the rest of the courses rather than being forced to select all available courses leaving the undesired ones.
  B. There will be a penalty for choosing an undesired course while no penalty is assigned for neglecting a desired one.

- Time periods is determines as one hour periods, although half an hour would be more efficient and comprehensive.
- It is unpractical for the user to fill the desirability degree for 75 period slots each time he uses the application.
- The application will assign a penalty for each assigned undesired hour, however, when the user specify a large period as undesired for some reason, this period should be penalized the same way whether it was violated entirely or partially.
- No constraints were built to handle the problem of the subordinated classes.
- No constraints were built to handle the problem of final exams conflicts.

Chapter 3: Background

3.1 The Credit Hours System

The credit hour system in higher education is considered a modern system that appeared in the late nineteenth century and was first adopted by the American and European universities in the early twentieth century. Now, most of the universities around the world use it.

Raubinger, Rowe, Piper, and West (1969) [9], described the history of the credit unit as divided into three phases:

- 1873–1908: Increasing dissatisfaction with the college admissions process and high school–to–college articulation
- 1908–1910: The proposal and implementation of a standard high school unit.
- 1910 to the present: The introduction of the Carnegie unit, its widespread growth, and its effect on both secondary and higher education

At the end of the 1800's and the beginning of 1900's, the Carnegie unit became the basis for granting high school diplomas and credit hours for the baccalaureate [10].

China has been using the scholastic year system since the 50s last century. Since 1978, Chinese universities have gradually adopted the credit system [11]. A semester credit hour is the most commonly used system of measuring course work and is usually based on at least a 14–17 week calendar [12].

In the credit hours system, a credit hour represents a measurement unit for courses that have to be studied by the student in order to fulfill the requirements of a certain major. It refers to the weekly teaching activity of a certain course during the semester such that a one hour course means that its classes meet for one hour a week.

In the credit hour system – adopted by the Islamic university of Gaza and most of the Palestinian universities –, the academic year is divided into two semesters, the first semester and the second semester with an optional summer semester. Each semester is usually 15 to 16 weeks except for the summer semester which is about 8 weeks. In each semester, the student usually takes between 12 to 18 hours or four to six courses. Each course is usually three credit hours which is three class hours a week for four months. However, the academic systems differs from a university to another, and sometimes from a college to another with respect to the maximum and minimum limits of credit hours allowed per semester or the number of credit hours assigned to courses.

3.1 Multi-Criteria Decision Making (MCDM)

A decision is a choice out of a number of alternatives. This choice is made in such a way that the preferred alternative is the "best" among the possible candidates. The decision maker does not only have the task to judge the performance of the alternatives in question under each criterion, he/she also has to weigh the relative importance of the criteria in order to arrive at a global judgment.

One must acknowledge the presence of several criteria which are at least partially contradictory and often non commensurable, leading to the development of MCDM.

MCDM is an advanced field of operations research that is devoted to the development and implementation of decision support tools and methodologies to confront complex decision problems involving multiple criteria, goals, or objectives of conflicting nature [13].

Numerous multi-criteria decisions are daily made, both in public and in private life. Such as a company choice of products and markets or the choice of a location for production. In private life, the choice of a partner or a career.

Methods for MCDM have been designed in order to designate a preferred alternative, to classify the alternatives in a small number of categories, and/or to rank the alternatives in a subjective order of preference; they may sometimes also be used to allocate scarce resources to the alternatives on the basis of the analysis results [14].

3.1.1 Steps of the MCDM process

The step of the MCDM process include

A. Problem identification

   The process of problem identification can be supported by stakeholders.

B. Defining relevant attributes

C. Extracting relevant criteria related to the attributes

   This step includes the building of the problem hierarchy.

D. Discussing and proposing alternatives

   Description of potential alternative actions for achieving the attributes.

E. Recognizing alternatives and eliminating the infeasible ones

   Design and execute the studies necessary to collect data for the decision criteria.

F. Making judgments and weighting the criteria-related preferences

   Elicit weighting structure for the criteria, as appropriate.

G. Building the decision matrix

   Populate a decision matrix for the alternatives and decision criteria.

H. Synthesizing and ranking alternatives

   Synthesize criteria and weights to rank alternatives.

I. Examining, verifying and documenting the decision

   Communicate with stakeholders and select alternatives [15].

3.2 Analytic Hierarchy Process (AHP)

AHP is one of MCDM methods; it was originally developed by Thomas L. Saaty in the mid-1970s. It combines tangible and intangible aspects to obtain the priorities associated with the alternatives of the problem.

AHP is a structural framework that allows decision-makers to model a complex problem in a hierarchical structure by breaking it down into smaller parts, then calling for a simple comparison with respect to pairs of judgments to develop priorities within each level of hierarchy. Finally, results are synthesized to obtain overall weights of the alternatives. The input can be obtained from actual measurement such as price, weight etc., or from subjective opinion such as satisfaction feelings and preference. AHP allows some small inconsistency in judgment because human is not always consistent. The ratio scales are derived from the principal Eigen vectors and the consistency index CI is derived from the principal Eigen values.

3.2.1 AHP Definition

According to Saaty definition (1977) "The AHP is a simple, mathematically based MCDM tool to model deal complex, unstructured and multi-attribute problems in a hierarchal structure showing the relationships of goal, criteria, sub criteria, and alternatives". AHP not only support decision makers by enabling them to structure complexity and exercise judgment, but it allows them to corporate both objective and subjective considerations on the decision problems.

3.2.2 AHP Principles and Axioms

AHP is built on a simple theoretical foundation to determine how much the alternatives contribute to the goal. According to Forman and Gass (2001), AHP is based on three basic principles; decomposition, comparative judgments and synthesis. The decomposition principle is applied to structure a complex problem into hierarchy of clusters, sub-clusters, sub- sub clusters and so on. The principle of comparative judgments is applied to construct pairwise comparisons of all combinations of elements in a cluster with respect to the parent of the cluster. The principle of synthesis or hierarchal composition is applied to multiply the priorities of elements in a cluster by the priority of the parent element.

Axioms provide the foundations for any methodology or technique. Saaty has specified four axioms for AHP and these have been described more simply by Forman and Gass (2001).

The first axiom; the reciprocal axiom, requires that if A is three times better than B, then B is one third as good as A.

The second axiom; the homogeneity axiom, states that the elements to be compared should not differ too much to not have large errors in judgments that lead to a decrease in accuracy and increase in inconsistency.

The third axiom states that the priorities of the elements in a cluster do not depend on lower level elements, that means when comparing elements at each level a decision-maker has just to compare with respect to the contribution of the lower-level elements to the upper-level one. This local concentration of the decision-maker on only part of the whole problem is a powerful feature of the AHP.

The fourth axiom; the expectation axiom, says that individuals who have reasons for their beliefs should make sure that their ideas are adequately represented for the outcomes to match these expectations. This axiom means that output priorities should not be radically different to any prior knowledge or expectation that a decision maker has [16].

3.2.3 AHP Methodology

AHP is based on the assumption that when faced with a complex decision, the natural human reaction is to cluster the decision elements according to their common characteristics. It involves building a hierarchy of decision elements and then making comparisons between each possible pair in each cluster. This gives a weighting for each element within a cluster and also a consistency ratio (CR) which is useful for checking the consistency of the data. The methodology of the AHP is explained in figure 3.1.

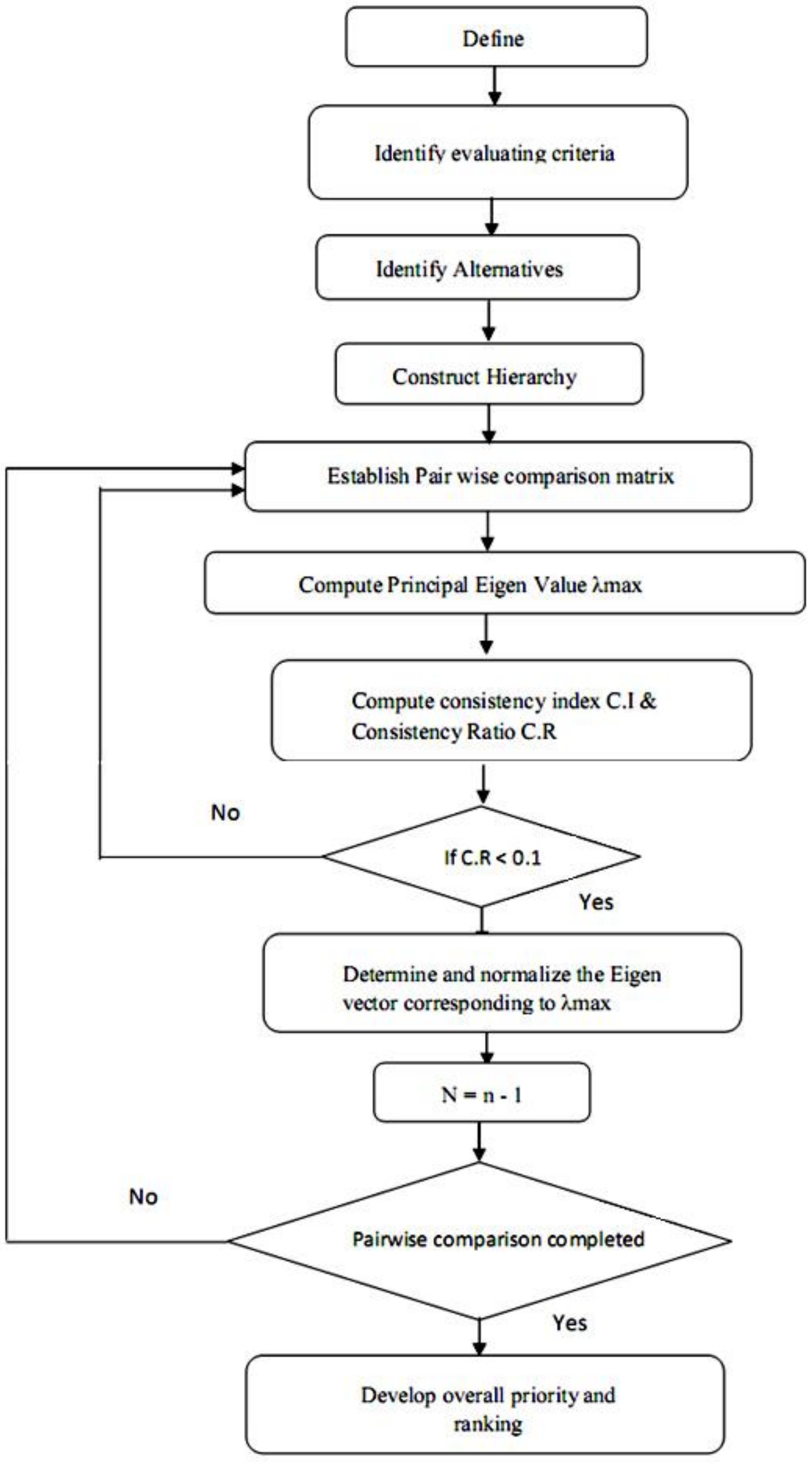

Define
Identify evaluating criteria
Identify Alternatives
Construct Hierarchy
Establish Pair wise comparison matrix
Compute Principal Eigen Value λmax
Compute consistency index C.I & Consistency Ratio C.R
No
If C.R < 0.1
Yes
Determine and normalize the Eigen vector corresponding to λmax
N = n - 1
No
Pairwise comparison completed
Yes
Develop overall priority and ranking

Figure 3.1: AHP Methodology (Saaty, 1980)

3.2.4 Hierarchical Structuring of the Problem

In the first stage, the decision maker defines a hierarchical structure representing the problem at hand. A general form of AHP structure is presented in figure (3.2). In the simplest case, the hierarchy has three levels. The first level represents the goal of the decision problem and is analyzed as resulting from the aggregation of evaluation criteria represented by the second level; the last level of the hierarchy involves the alternatives to be evaluated. In more complex cases, there may be more levels, corresponding to splitting criteria into sub-criteria. The objective or the overall goal of the decision is represented at the top level of the hierarchy. The criteria and sub-criteria contributing to the decision are represented at the intermediate levels. Finally, the decision alternatives or selection choices are laid down at the last level of the hierarchy. The number of the levels in a hierarchy depends on the complexity of the problem being analyzed and the degree of detail of the problem that an analyst requires to solve.

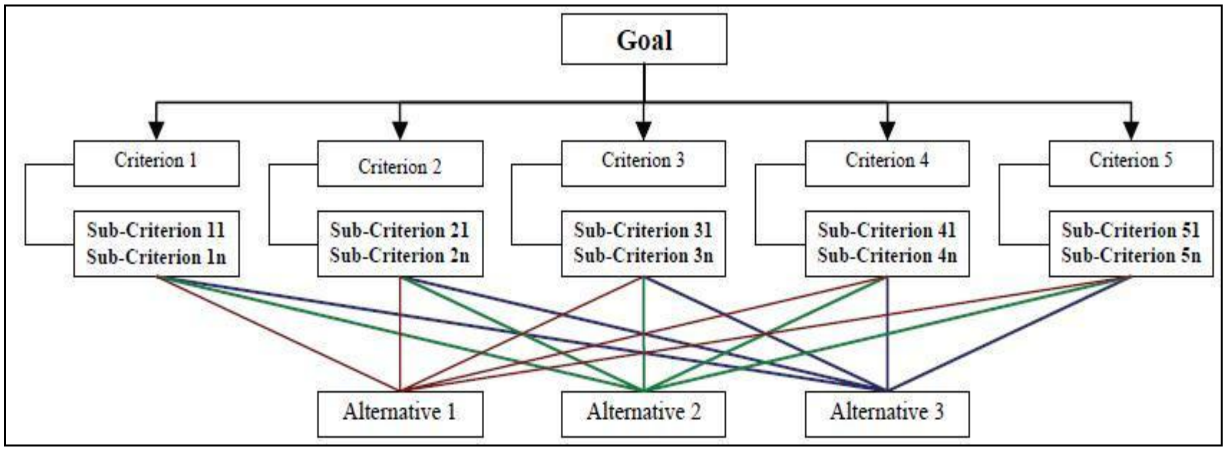


Figure 3.2: AHP Hierarchy (Saaty, 1980)

3.2.5 Performing Pairwise Comparisons

Once the hierarchy of the problem is defined, the decision-maker performs a series of pairwise comparisons within the same hierarchical level and then between sections at a higher level in the hierarchy structure to have n*(n−1)/2 comparisons if there are n criteria. In comparisons, a ratio scale of 1−9 is used to compare any two elements. Table (3.1) shows the measurement scale defined by Saaty (1980). The matrix of pair-wise comparisons is:

$$\underline{A} = \begin{bmatrix} w_1/w_1 & w_1/w_2 & w_1/w_3 & \cdots & w_{1/}w_n \\ w_2/w_1 & w_2/w_2 & w_2/w_3 & \cdots & w_2/w_n \\ w_3/w_1 & w_3/w_2 & w_3/w_3 & \cdots & w_3/w_n \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ w_n/w_1 & w_n/w_2 & w_n/w_3 & \cdots & w_n/w_n \end{bmatrix}$$

Figure 3.3: Pairwise Comparison Matrix

Table (3.1): Saaty's Scale of Importance Intensities [Saaty, 1980].

Intensity of importance	Definition
1	Equal importance
3	Weak importance of one over another
5	Essential or strong importance
7	Demonstrated importance
9	Absolute importance
2,4,6,8	Intermediate values between the two adjacent judgments

The pairwise comparisons of various criteria are organized into a square matrix as shown in matrix A. The diagonal elements of the matrix are 1. The criterion in the ith row is better than criterion in the jth column if the value of element (i, j) is more than 1; otherwise the criterion in the jth column is better than that in the ith row. The (j, i) element of the matrix is the reciprocal of the (i, j) element. The pair wise comparisons depend on subjective judgment without any scientific measurements, so it has been verified that a number of these pairwise comparisons taken together forms a sort of average. This average is calculated through a complex mathematical process using Eigen values and Eigen vectors. The principal Eigen value and the corresponding normalized right Eigen vector of the comparison matrix give the relative importance of the various criteria being compared. The elements of the normalized Eigen vector are termed weights with respect to the criteria or sub-criteria and ratings with respect to the alternatives (Saaty, 1980).

The procedure of pairwise comparison is to evaluate the importance of the criteria and then the preference for the alternatives with respect to each criterion.
The final solution results in the assignment of weights to the alternatives located at the lowest hierarchical level.

3.2.6 Synthesis

Once judgments have been entered for each part of the model, the rating of alternative is multiplied by the weights of the sub-criteria and aggregated to get local ratings with respect to each criterion. The local ratings are then multiplied by the weights of the criteria and aggregated to get global ratings. The AHP produces weight values for each alternative based on the judged importance of one alternative over another with respect to a common criterion. The results are then synthesized to obtain rank of the alternatives in relation to the overall goal.

3.2.7 Consistency Evaluation

Comparisons made are subjective and AHP tolerates inconsistency through the amount of redundancy in the approach. If this CI fails to reach a required level, then answers to comparisons may be re-examined. The Eigen value technique enables the computation of a consistency measure which is an approximate mathematical indicator of the inconsistencies or intransitivity in a set of pairwise ratings. This consistency measure is called the CI which is calculated as:

$$CI= (\lambda \max-n)/ (n-1)$$

Where λmax is the maximum Eigen value of the judgment matrix.

This CI can be compared with that of Random Consistency Index, (RI). RI can take a value between 0 to 1.49 as shown in table (3.2). The ratio derived, CI/RI, is termed the CR, Saaty suggests the value of CR should be less than 0.1, if it is greater than 0.1 (or 10%), the level of inconsistency in the set of ratings is considered to be unacceptable. In this situation, the evaluation procedure has to be repeated to improve consistency. Sensitivity analysis can be performed to see how well the alternatives performed with respect to each of the objectives as well as how the alternatives are sensitive to changes of the objectives. (Saaty, 1980)

Table (3.2): Random Consistency Index (RI) [Saaty, 1980].

N	1	2	3	4	5	6	7	8	9	10
RI	0	0	0.58	0.9	1.12	1.24	1.32	1.41	1.45	1.49

3.2.8 AHP Applications

Broad areas where AHP has been successfully employed include: selection of one alternative from many; resource allocation; forecasting; total quality management; business process re-engineering; quality function deployment, and the balanced scorecard (Saaty and Vargas, 1991). By scanning the literature different uses of AHP can be found these include:

- Serkan et al. (2009) used AHP and TOPSIS methods under fuzzy environment for weapon selection.
- Hambali et al. (2009) applied AHP for composite manufacturing process selection.
- Steven (2008) used AHP for asset allocation.
- Agha (2008) used AHP for evaluating and benchmarking non-governmental training programs.
- Ahmet and Bozbura (2007) used AHP for prioritization of organizational capital measurement indicators.
- Forman and Gass (2001) constructed AHP model for assessing risk in operating cross-country petroleum pipelines.
- Babic and Palzibat (1998) used AHP for ranking of enterprises according to the achieved level of business efficiency.
- Berrittella, (2007) used AHP in deciding how best to reduce the impact of global climate change
- McCaffrey, (2005) used AHP in quantifying the overall quality of software systems in Microsoft Corporation
- Grandzol, (2005) used AHP in selecting university faculty in Bloomsburg University of Pennsylvania.

- Atthirawong, (2002) used AHP in deciding where to locate offshore manufacturing plants.
- Dey, (2003) used AHP in assessing risk in operating cross-country petroleum pipelines for American Society of Civil Engineers.
- Chengjing Jounio (2013) used AHP to find the best suitable supplier in China.
- Zimmer et al. (2012) used AHP to evaluate projects.

3.2.9 AHP Strengths and Weaknesses

Several researchers, including Triantaphyllou and Mann (1990), have pointed out the weakness of AHP as follows

A. Weaknesses of AHP

- **High inconsistency Ratio (CR) Between the Stakeholders**

The weakness of AHP in assessing the relative importance weights of various criteria, in addition to that the ability of humans to accurately express their knowledge decreases with increasing problem complexity, are considered the two main sources of the high inconsistency ratio (CR). The weakness in assessing the relative importance weights of various criteria results primarily from two limitations, the difficulty of using Saaty's discrete 9-value scale to reflect the belief of decision makers in the relative importance relationship among the various criteria, and the difficulty of identifying the in-between numbers of fuzzy sets. Saaty's discrete 9-value scale method forces decision makers to select numbers from the finite set {1/9, 1/8, 1/ 7... 1, 2, 3... 7, 8, 9}, contradicting the real world fuzzy memberships of elements in a fuzzy set. In most real world problems, the membership values in a fuzzy set take on continuous values (namely real numbers) rather than discrete numbers. Triantaphyllou and Mann, (1990), found that this limitation can cause extremely high failure rates for AHP.

- **Rank Reversal**

Other drawback sometimes arises with AHP known as ‘rank reversal’, which is associated with the relative nature of the judgments involved. Here, changing the set of alternatives changes the ranking of all alternatives. If new alternatives are likely to be added to the model after initial analysis, and alternatives are amenable to a direct rating approach (i.e. not so qualitative as to require pair wise comparison), then an approach in which ratings of alternatives are assigned directly (such as the Simple Multi-Attribute Rating Technique or SMART) could be a better choice.

- **Complexity**

AHP is by nature a multi-stakeholder and multi criteria approach to decision-support. Such feature may make using AHP especially for strategic decision making. The first obstacle faced while dealing with such case is lack of agreement on how to identify stakeholder groups, and how to select samples or representatives from them. Stakeholders' interviews sometimes are long. So, a well-trained stuff is needed to prepare a valid questionnaire as well as explain the questions briefly and obviously. In short, AHP may appear invalid approach in situations where time is crucial.

On the contrary, according to Morrissey and Browne (2004), a number of benefits have been noted with the AHP process in general as a (MCDM) technique.

B. AHP Strengths

AHP has been applied in a wide variety of decision areas including those related to economy, planning, energy policy, health, conflict resolution, project selection, budget allocation (Zahidi, 1985), operations management (Partovi et al., 1990), benchmarking (Eyrich, 1991), total quality management, win-win management (Gunther et al., 2002), site selection, and education (Bahurmoz, 1999 & 2003). In addition to being used alone, the AHP has been combined with a number of quantitative analysis techniques such as LP, goal programming, Data Envelopment Analysis, game theory, conjoint analysis and SWOT analysis (ISAHP 1999 & 2001).

Narasimhan (1983) states the benefits of using AHP as follows:

- It formalizes and makes systematic what is largely a subjective decision process and thereby facilitates "accurate" judgments.
- As a by-product of the method, management receives information about the evaluation criteria's implicit weights.
- The use of computers makes it possible to conduct sensitivity analysis of the results.

Wu & Wu (1984) adapted the AHP technique for the selection of the best single plant location reported the benefits of it as follows:

- AHP is an effective management tool. It can handle many alternatives at one time and so permit comparisons to be made. Other popular techniques, such as the Relative Merit Method or Dimensional Analysis, can only handle two alternatives at a time.
- The AHP can handle complex situations where different weights are assigned to the same attributes. Judges' opinions may vary when determining how important an attribute is. Also, a weight could be assigned to the Judges' authority in the decision-making process. For instances, the President of a firm may have more

say than the Vice President. Therefore, his opinion can be weighted at 0.65 and the Vice President's at 0.35. This rationale could also be applied to several stockholders.

- **Inclusiveness**

A mixture of quantitative and qualitative information and taking into account multiple stakeholders with conflicted objectives makes AHP to go beyond the evaluation of purely economic consequences and allows non-economic criteria to be assessed on an equal basis, which enhance the results confidence.

- **Flexibility**

The hierarchal nature of AHP makes priority of each element depend on the higher level elements. So, if the surrounded conditions lead to change the judgment of any criteria the final rank of the alternatives will change according to the changes in the ground. So, managers can automatically allocate their resources to accommodate the new circumstances.

- **Easiness**

AHP methodology does not depend on cumbersome mathematical concepts. So, it is easy to understand and be applied by the majority of people. AHP easiness makes it one of the most decision making widely used tools. In addition to all AHP benefits and drawbacks were mentioned above, the following table, Table (3.3), summarizes other pros/ cons related to it.

Table 3.3: Pros and cons of AHP [17]

Pros	Cons
• It allows MCDM. • It is applicable when it is difficult to formulate criteria evaluations, i.e., it allows qualitative evaluation as well as quantitative evaluation. • It is applicable for group decision making environments • The inclusion of the managers at every step of the decision analysis in the AHP method gave them a feeling of ownership that nearly insured the implementation of the findings. • Inconsistency measure helps users to know when they make inconsistent judgments, especially if they are working as a group. People want to be logically consistent in making decisions. • Using AHP in group setting results in better communication, leading to clearer understanding and consensus among the members of decision making group, and hence a greater commitment to the chosen alternative.	• There are hidden assumptions like consistency. Repeating evaluations is cumbersome. • Difficult to use when the number of criteria or alternatives is high, i.e., more than 7. • Difficult to add a new criterion or alternative • Difficult to take out an existing criterion or alternative, since the best alternative might differ if the worst one is excluded.

3.3 Goal Programming (GP)

One of the most optimistic techniques for multiple objective decision analysis is GP. This is a powerful tool which draws upon the highly developed and tested technique of LP but it also provides a simultaneous solution to a complex system of competing objectives. GP can handle decision problems having a single goal with multiple sub-goals.

Generally, many decision problems in organizations involve multiple objectives. Such problems are not simple to analyze by optimization techniques such as LP. (MCDM) or multiple-objective decision making (MODM) has been a popular topic of management science during the past decade. A number of different approaches of MCDM or MODM have been proposed, such as the multi-attribute utility theory, the multiple-objective "LP", "GP", "Compromised Programming" and various heuristics. Among these, "GP" has been the most widely accepted and applied technique [18].

Ijiri (1965), stated that "In conventional LP the objective function is one-dimensional, intended either to maximize effectiveness or to minimize sacrifice. GP techniques are capable of handling multiple goals in multiple dimensions and therefore have no dimensional limitation of the objective function.

GP techniques offer optimal solutions to the problem of conflicting or incommensurable goals if an ordinal ranking of goals in terms of their contributions or importance to the organization can be provided [19].

3.3.1 Goal Programming Definition

Goal Programming is a branch of multiple objective programming, which in turn is a branch of MCDA, also known as MCDM. It can be thought of as an extension or generalization of linear programming to handle multiple, normally conflicting objective measures. Each of these measures is given a goal or target value to be achieved. Unwanted deviations from this set of target values are then minimized in an achievement function. This can be a vector or a weighted sum dependent on the GP variant used [20].

Rifai (1994), defined in GP as "Mathematical model manages a set of conflicting objectives by minimizing deviations between the target values and the realized". An explicit definition of GP was given by Charnes and Cooper (1961) as "a branch of multi objective optimization that can be thought of as an extension or generalization of LP to handle multiple, normally conflicting objectives."

GP handles the MCDM problems through considering the measures related to the conflicting objectives as a given goal or target value to be achieved. Unwanted deviations from this set of target values are then minimized in an achievement function. This can be a vector or a weighted sum dependent on the GP variant used. As satisfaction of the target is deemed to satisfy the decision-maker(s), an underlying satisfying philosophy is assumed.

GP is a well-known modification and extension of LP. LP deals with only one single objective to be minimized or maximized, and subject to some constraint; therefore, has limitations in solving a problem with multiple objectives. GP, instead, can be used as an effective approach to handle a decision concerning multiple and conflicting goals. Further, the objective function of a GP model may consist of non-homogeneous units of measure [19].

3.3.2 History of GP

GP was first used by Charnes, Cooper and Ferguson in 1955, although the actual name first appear in a text by Charnes and Cooper (1961). Seminal works by Lee (1972), Ignizio (1976), Ignizio and Cavalier (1994) and Romero (1991) followed. Scniederjans (1995) gives in a bibliography of a large number of pre 1995 articles relating to GP and Jones and Tamiz give an annotated bibliography of the period (Jones and Tamiz, 1990-2000). The first engineering application of GP, due to Ignizio in 1962, was the design and placement of the antennas employed on the second stage of the Saturn V. This was used to launch the Apollo space capsule which landed the first men on the moon [20].

GP is a branch of MCDA. It was first introduced by Charnes et al. in 1955, more explicitly defined by the same authors in 1961, and further developed by Ijiri during the 1960's. The first books dedicated to GP by Lee and Ignizio appeared during the early to mid 1970's. In the 1970's, GP and its variants were applied to many different subject areas.

Questions were raised as to the effectiveness of GP as an application tool by Zeleny and Harrald during the late 1970's and early 1980's, but GP still grew in popularity judging by the increase of papers applying GP during that period [21].

3.3.3 GP Algorithms

Three basic methods have been developed to optimize a multi objective model with possibly conflicting goals:

a. The weights method (mini–sum)

b. The Lexicographic method (preemptive)

c. Chebyshev GP (mini–max)

1. Weighted GP (WGP)

The objective is to find a solution that minimizes the weighted sum of the goal deviations. If the decision–maker is more interested in direct comparisons of the objectives then weighted, or non pre–emptive, GP should be used. In this case all the unwanted deviations are multiplied by weights, reflecting their relative importance, and then added together as a single sum in order to minimize the weighted sum of the goal deviations. It is important to recognize that deviations measured in different units cannot be summed directly due to the phenomenon of incommensurability. Hence each unwanted deviation is multiplied by a normalization constant to allow direct comparison. Popular choices for normalization constants are the goal target value of the corresponding objective (hence turning all deviations into percentages) or the range of the corresponding objective between the best and the worst possible values, hence mapping all deviations onto a zero–one range [22].

WGP assumes that the positive and negative deviations of the criterion outcomes are equally undesirable. That is, that decision–maker perceives both overachievement and underachievement of specified goals as equally undesirable outcomes.

Chang (2007) defined the WGP structure in the following model:

$$Min\, Z = \sum_{i=1}^{n}(\alpha_i\, d_i^+ + \beta_i\, d_i^-) \tag{3.1}$$

Subject to:

$$f_i(x) - d_i^+ + d_i^- = g_i \quad i = 1,2, \ldots .n \quad i \in h_r \tag{3.2}$$

$$d_i^+, d_i^- \geq 0 \quad i = 1,2, \ldots .n \tag{3.3}$$

$$X \in F \ (F\ is\ feasible\ set)$$

Where

$f_i(x)$: is the linear function of the ith goal.

g_i: is the aspiration level of the ith goal.

h_r: represent the index set of goals placed in the rth priority level.

αi and βi: are the respective positive weights attached to these deviations in the achievement function.

$d_i^+ = \max(0, f_i(x) - g_i)\ and\ d_i^- = \max(0, g_i - f_i(x))$ respectively, over and under achievements of the ith goal.

2. Lexicographic GP (LGP)

The initial GP formulations ordered the unwanted deviations into a number of priority levels, with the minimization of a deviation in a higher priority level being of infinitely more importance than any deviations in lower priority levels. This is known as lexicographic or pre-emptive GP. Ignizio (1976) gives an algorithm showing how a lexicographic GP can be solved as a series of LP. LGP should be used when a clear priority ordering exists amongst the goals to be achieved. Chang (2007) defined the LGP structure in the following model:

$$Min\ Z = [\ \sum_{i \in h1}\ (\alpha_i\ d_i^+ +\ \beta_i\ d_i^-), \dots \dots \sum_{i \epsilon hr}\ (\alpha_i\ d_i^+ +\ \beta_i\ d_i^-)] \tag{3.4}$$

Subject to:

$$f_i(x) -\ d_i^+ + d_i^- = g_i\ i = 1,2, \dots .n\ i \in h_r \tag{3.5}$$

$$d_i^+, d_i^- \geq\ 0 \quad i = 1,2, \dots .n \tag{3.6}$$

$$X\ \in F\ (F\ is\ feasible\ set)$$

Where definitions of variables included in this model (LGP) are the same as the (WGP). Objective functions are ordered according to their importance. Given the ordering, the most important function is minimized first, then on the set of optimal solutions with respect to the first function the second function is minimized, and so on, until a unique solution is obtained or all the specified functions are minimized. This implies that goals of higher priority must be met before those of lower priority are considered.

3. Chebyshev GP (CGP)

Can be considered a specific form of a WGP approach, it seeks the solution that minimizes the worst unwanted deviation from any single goal. For decision-makers more interested in obtaining a balance between the competing objectives, CGP, which is considered a specific form of a WGP approach, should be used. Introduced by Flavell, (1976), this variant seeks to minimize the maximum unwanted deviation, rather than the sum of deviations. This utilizes the Chebyshev distance metric, which emphasizes justice and balance rather than ruthless optimization. Chang (2007) defined the CGP structure in the following model:

$Min\, Z$

Subject to:

$$Z \geq \alpha_i \, d_i^+ + \beta_i \, d_i^- \tag{3.7}$$

$$f_i(x) - d_i^+ + d_i^- = g_i \; i = 1,2,\ldots.n \; i \in h_r \tag{3.8}$$

$$d_i^+, d_i^- \geq 0 \quad i = 1,2,\ldots.n \tag{3.9}$$

$X \in F \; (F \; is \; feasible \; set)$

Where

Z: is an extra continuous variable that measures the maximum deviation. While definitions of variables included in this model (CGP) are the same as the (WGP).

Instead of using subjective notions to set the aspiration levels for the objectives, a set of single optimization problems is solved to arrive at the “best” and “worst” possible values of each objective. The best values are then used as aspiration levels for the objectives. The objective then becomes to minimize the deviation from those aspiration levels so that the worst deviation from any single–goal aspiration level is minimized [23].

3.3.4 GP strengths and weaknesses

A. GP weaknesses

In spite of the vital role of GP in handling the problem with multi criteria and multi stakeholders; GP methodology suffers from some limitations that need to be overcome to enhance its ability to give more accurate and confident decisions. The following sections summarize the limitations of GP.

- **Incommensurability**

Incommensurability means the incompatibility of different decision variables into a single objective function, which mainly occurs due to the use of different units of deviational variables in an objective function of weighted goal programming where the sum of unwanted deviational variables are minimized. These different measurement units damage the relative importance of the objective to the decision maker (Tamiz and Jones, 1994).

- **Complexity**

Making decisions is part of our daily lives. In fact, the conflicts of resources and the incompleteness of available information make it almost impossible for decision-makers to build a reliable mathematical model for representation of their preferences. In order to overcome the problem of underestimation of the decision, the decision-makers according to the above mentioned, not only must consider the only single aspiration level in the local region, but also develop multiple aspiration levels under given constraints to obtain more confident solution. It is obvious that the complexity of the Multi-Criteria GP (MCGP) problem with n aspiration levels requires adding (ln n /ln 2) extra binary variables. The proposed GP model, with membership function, is used to handle the MODM problem with imprecise aspiration levels of the proposed. Multi-Choice Aspiration Levels (MCAL) model is used for solving the MODM problem with more than one aspiration level [24].

- **Sensitivity**

The results obtained by solving the model, the model output, are completely dependent on the importance weights. So, GP requires that the decision-maker specify fairly detailed a priori information about his or her aspiration levels, preemptive priorities, and the importance of goals in the form of weights.

In many complex problems, it is difficult (or even impossible) for the decision maker to provide the precise information required by these methods, these difficulties are aggravated further when the goals are unrelated to each other.

McGeehan (1978) listed the primary disadvantages of GP in its linear form as follows:

A. The objective function or achievement function, constraints and goal relations must be linear. In fact, true linearity may not exist. GP requires that the measure of goal attainment and resource utilization be proportional to the level of each activity conducted individually.
B. Fractional values of decision variables must be acceptable in the solution because the optimal solution of a linear GP problem often yields non-integer values for the decision variables.
C. GP requires a static rather than a dynamic environment. This due to the fact that the model coefficients must be constants rather than subject to change as conditions change. This disadvantage can be minimized by including in the model coefficients which are based on forecasts of future conditions [25].

D. GP strengths

Despite the existence of some limitations related to the GP model, GP has enough strengths to be considered one of the most important multi objectives mathematical programming models. These strengths are mentioned in the following sections.

• Simplicity

A major strength of GP is its simplicity and ease of use. This accounts for the large number of GP applications in many and diverse fields. As weighted and CGP can be solved by widely available LP computer packages, finding a solution tool is not difficult in most cases. LGP can be solved as a series of LP models, as described by Ignizio and Cavalier (1994).

- **Flexibility**

The weights, aspiration levels, preemptive priorities can be changed during the analysis as the decision maker's knowledge of the decision problem changes (Interactive Programming). So, when it is necessary to change the model's input according to the business rapid change nature, no much efforts are needed for modify the model construction to be suitable for the new scenarios.

McGeehan (1978) listed the primary advantages of GP over traditional decisions processes as follows:

- It helps define the decision environment in ambiguous terms.
- It provides systematic consideration of alternative decision strategies, often involving different levels of management.
- It ensures that all key elements are considered each time a decision strategy is evaluated.
- It creates a documented record of the decision process.
- It provides quantitative solutions to management problems.

On balance, the advantages of GP appear to outweigh the disadvantages for the problems of decision making [25].

3.4 Combined AHP and GP

GP is a structured decision-making approach used to evaluate and satisfying solution based on the priorities or weighted ranking assigned to each goal. While

GP provides no systematic method to prioritize or rank relative importance or weights of the goals, the AHP measures the relative importance of multiple goals with consistency. A systematic approach to rank elements (goals or alternatives) in AHP can be utilized in the replacement of a subjective judgment to prioritize each goal in GP. Khorramshagol and Ignition (1984) originally discussed an integration of GP and AHP concepts in the study of single and multiple decision- making in a multiple objective environment.

Since AHP is most widely accepted remedy to establish a relative importance among goals, the integrated model in the study utilizes AHP to determine the priorities to be used in GP model development to solve the problem.

The use of AHP alone for a strategic selection problem is not sufficient, because it is not able to incorporate the resource constraints, dependencies among the alternatives and multiple conflicting goals, criteria, and sub criteria into their decision structure.

At the same time, GP cannot also be used alone, because it still requires calculation of the weights of various criteria to use in the objective function of the GP model. One of the most suitable solutions of this dilemma is to use a combination of (AHP) with GP in order to gain a final solution that is nearest to the ideal one. Table 3.4 presents examples of studies which approach depended on Combined AHP and GP.

Table 3.4: Combined AHP–GP application from literature [26]

Authors	Applications	Specific areas
Schniederjans and Garvin (1997)	Business	Cost driver selection
Kwak and Lee (1998)	Higher education	IT–based project selection
Radash and Kwak (1998)	Marketing	Offset proposal selection
Badri (1999)	Logistics	Facility location selection
Guo and He (1999)	Agriculture	Harvesting measure selection
Kim et al. (1990)	Military	Nuclear fuel cycle selection
Zhou et al. (2000)	Health Care	IT–based project selection
Badri (2001)	Logistics	Scheduling plan selection
Kwak and Lee (2002)	Service	Customer data collection method selection
Radeliffe and schniederjans (2003)	Health care	IT–based project selection
Wang et al. (2004)	Industry	Trust factor selection
Yurdakul (2004)	Logistics	Supplier selection
Kwak et al. (2005)	Manufacturing	Computer–integrated manufacturing technology selection
Wang et al. (2005)	Marketing	Advertising medium selection
Bertolini and Bevilacqua (2006)	Logistics	Supplier selection
Slah Bahloul and Fathi Abid (2013)	Business	International portfolio selection in the presence of investment barriers.

3.5 Decision Support Systems (DSS)

3.5.1 Definitions

The concept of a decision support system (DSS) is extremely broad and its definitions vary depending on the author's point of view [27]. It can take many different forms and can be used in many different ways [28]. On the one hand, Finlay (1994) and others [29] define a DSS broadly as "a computer-based system that aids the process of decision making". In a more precise way, it can be defined as "an interactive, flexible, and adaptable computer-based information system, especially developed for supporting the solution of a non-structured management problem for improved decision making. It utilizes data, provides an easy-to-use interface, and allows for the decision maker's own insights." [30]. For Keen and Scott Morton (1978) [31], "DSS are computer-based support for management decision makers who are dealing with semi-structured problems." For Sprague and Carlson (1982) [32], DSS are "interactive computer-based systems that help decision makers utilize data and models to solve unstructured problems."

A DSS may be defined by its capabilities in several critical areas-capabilities which are required to accomplish the objectives which are pursued by the development and use of a DSS. Observed characteristics of a DSS which have evolved from the work of Alter (1977), Keen (1977), and others include:

- They tend to be aimed at the less well structured, underspecified problems that upper level managers typically face;
- They attempt to combine the use of models or analytic techniques with traditional data access and retrieval functions;
- They specifically focus on features which make them easy to use by non-computer people in an interactive mode.

- They emphasize flexibility and adaptability to accommodate changes in the environment and the decision making approach of the user.

The manager or user is the person faced with the problem or decision – the one that must take action and be responsible for the consequences [33].

3.5.2 Three DSS technology levels

It is helpful to identify three levels of hardware/software which have been included in the label "DSS." They are used by people with different levels of technical capability, and vary in the nature and scope of task to which they can be applied.

1. Specific DSS

The system which actually accomplishes the work might be called the Specific DSS. It is an information systems "application," but with characteristics that make it significantly different from a typical data processing application. It is the hardware/software that allows a specific decision maker or group of decision makers to deal with a specific set of related problems.

2. DSS Generator

The second technology level might be called a DSS Generator. This is a "package" of related hardware and software which provides a set of capabilities to quickly and easily build a Specific DSS.

3. DSS Tools

The third and most fundamental level of technology applied to the development of a DSS might be called DSS Tools. These are hardware or software elements which facilitate the development of a specific DSS or a DSS Generator. This category of technology has seen the greatest amount of recent development, including new special purpose languages, improvements in operating systems to support conversational approaches, color graphics hardware and supporting software, etc [33].

The relationships between these three levels of technology and types of DSS are illustrated by figure 3.4.

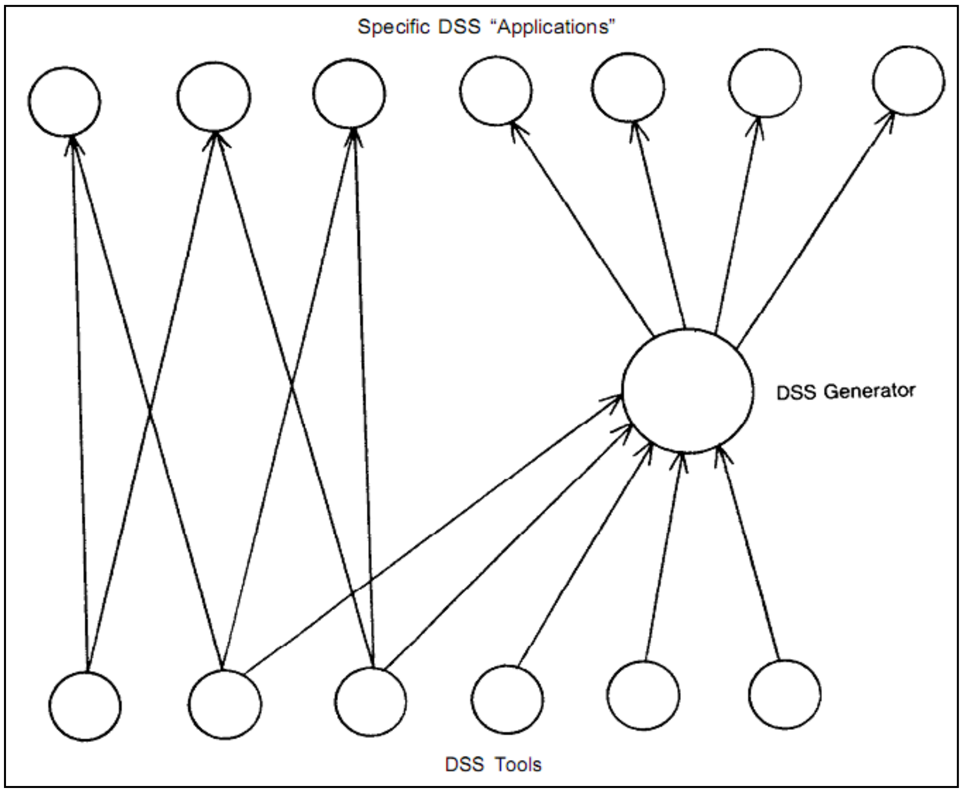


Figure 3.4: **Three levels of DSS Technology**

Components of a specific DSS

Three fundamental components of a DSS architecture are:

1. The database (or knowledge base),
2. The model (i.e., the decision context and user criteria), and
3. The user interface.

The users themselves are also important components of the architecture [34].

3.5.3 DSS Classification

Different authors propose different classifications. Using the relationship with the user as the criterion, Haettenschwiler (1999) [34] differentiates passive, active, and cooperative DSS. A passive DSS is a system that aids the process of decision making, but that cannot bring out explicit decision suggestions or solutions. An active DSS can bring out such decision suggestions or solutions which is the case in this DSS. A cooperative DSS allows the decision maker (or its advisor) to modify, complete, or refine the decision suggestions provided by the system, before sending them back to the system for validation. The system again improves, completes, and refines the suggestions of the decision maker and sends them back to him for validation. The whole process then starts again, until a consolidated solution is generated.

Another taxonomy for DSS has been created by Daniel Power. Using the mode of assistance as the criterion, Power differentiates communication-driven DSS, data-driven DSS, document-driven DSS, knowledge-driven DSS, and model-driven DSS [35].

- A communication-driven DSS supports more than one person working on a shared task; examples include integrated tools like Microsoft's NetMeeting or Groove [36]

- A data-driven DSS or data-oriented DSS emphasizes access to and manipulation of a time series of internal company data and, sometimes, external data.
- A document-driven DSS manages, retrieves, and manipulates unstructured information in a variety of electronic formats.
- A knowledge-driven DSS provides specialized problem-solving expertise stored as facts, rules, procedures, or in similar structures [35].
- A model-driven DSS emphasizes access to and manipulation of a statistical, financial, optimization, or simulation model. Model-driven DSS use data and parameters provided by users to assist decision makers in analyzing a situation; they are not necessarily data-intensive. Dicodess is an example of an open source model-driven DSS generator [37].

An optimization-based DSS – the one built here - can be classified as a model-driven DSS since it generates linear programming models.

3.5.4 DSS Benefits

1. Improves personal efficiency.
2. Speed up the process of decision making.
3. Increases organizational control.
4. Encourages exploration and discovery on the part of the decision maker.
5. Speeds up problem solving in an organization.
6. Facilitates interpersonal communication.
7. Promotes learning or training.
8. Generates new evidence in support of a decision.
9. Creates a competitive advantage over competition.
10. Reveals new approaches to thinking about the problem space.
11. Helps automate managerial processes.
12. Create Innovative ideas to speed up the performance.

Chapter 4: System Design

This chapter will focus mainly on the practical part of the research. Steps followed to build the DSS will be described in details. The theoretical design of the system will be the first section of the chapter in which system idea and structure will be presented. Moreover, the theoretical formulation of the standard model that is supposed to be generated by the system will be illustrated, beginning with the definition of the variables, the objective function and going through the various soft and hard constraints. The following part will focus on the software that is designed to finally, assumptions used in formulation.

In addition, the approach followed to incorporate the analytic hierarchy process is viewed.

The second section of this chapter presents the developed computer software. The actual code of the software will not be presented, however, its various parts will be presented in details clarifying all of the controls and tools used in them, meanwhile, some programming aspects regarding some logical issues will be discussed. Finally, guidance on steps that should be followed to properly use the software will be introduced.

The third and the last section of the chapter discuss theoretically the information feeding mechanism that is supposed to be developed by universities that wish to adopt the system. This mechanism may and may not be actually implemented through this research due to differences in programming languages and/or databases management systems, however, a general approach of it will be given. Besides, an attempt will be done in the Islamic university of Gaza with the help of the registration program specialists to implement this mechanism.

4.1 The Theoretical Design of the System

The idea of the system is to maximize achievement of a student registration goals using integer goal programming while avoiding all kinds of conflicts, it will provide the student with a ready, quick and optimal registration with respect to his/her personal preferences. A satisfactory level of these goals will be reached according to the priorities set by the student and to the extent that the available and suitable courses for him in that semester allow.

The core part of the DSSPS is a computer software which acts as a model generator. As soon as a user log in to the software, it will contact the university database and query about the current available courses and sections for that user. The software will then take input from the user about his/her criteria of interest and the associated importance weight of each generated by the analytical hierarchy process incorporated in the software - as pairwise comparisons between the different criteria are supposed to be set by the user - then, based on the enquired data, the software will generate an integer multi-objective optimization model that can be solved using one of the various linear programming engines to come out with the optimal solution from the student perspective and according to his specified criteria and weights while taking responsibility of overstepping all kinds of conflicts. The resulting solution represented in zeros and ones will go back to the software to be translated to a readable schedule form. (Figure 4.1) illustrates the structure of the DSSPS while figure 4.2 illustrates IPO schema of the system.

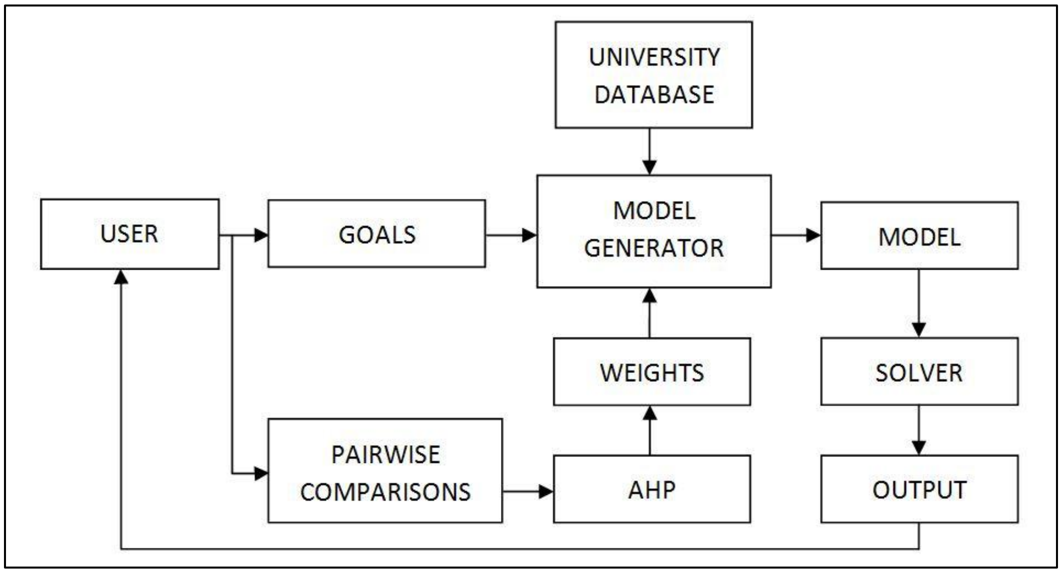


Figure 4.1: DSSPS Structure

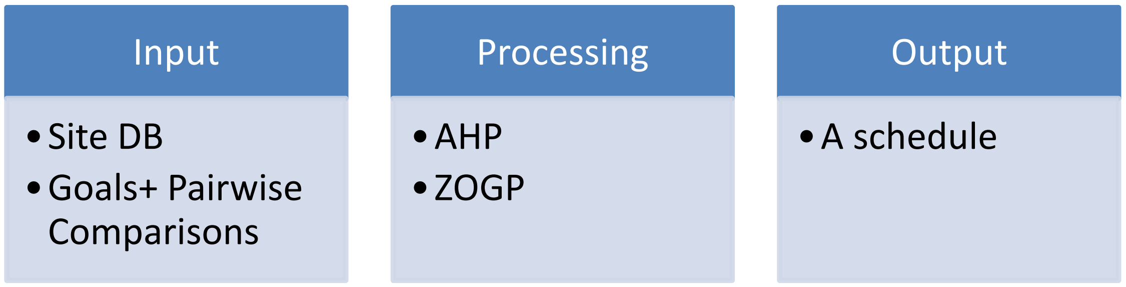


Figure 4.2: IPO schema of the system

The approach used to develop this system integrated both operation research and computer science. Operation research is represented in modeling techniques and optimization algorithms while computer science is represented in databases management, graphical user interface (GUIs), and software development techniques.

The approach used in model construction depends mainly on the software enquired database. The main idea of this DSS is to exploit the way by which programming languages deal with databases to the benefit of the model formulation and generation. The database is designed in such a way that each row in any of its tables consists of a group of fields that represent different characteristics of either the course or the section, thus, a decision variable is declared to represent each row in the database, consequently, giving it the value of one means the acceptance of all associated characteristics.

4.1.1 Database

As for now, the system still in the developing stage, because of that, this system was designed to operate with a local Access backend database for the purpose of developing and testing.

The software backend database should be formed to contain information about courses and their classes suitable for a student to be registered in a certain semester. This database consists of two tables, the available courses table and the available sections table. For a course to be considered "Available", it should meet the following conditions:

a. Offered for the current semester.
b. Unstudied before.
c. Their prerequisites are fulfilled.
d. At least, one section of it still available. Any course of which all sections are full should not be listed in the courses table, even though it satisfies all three previous conditions.

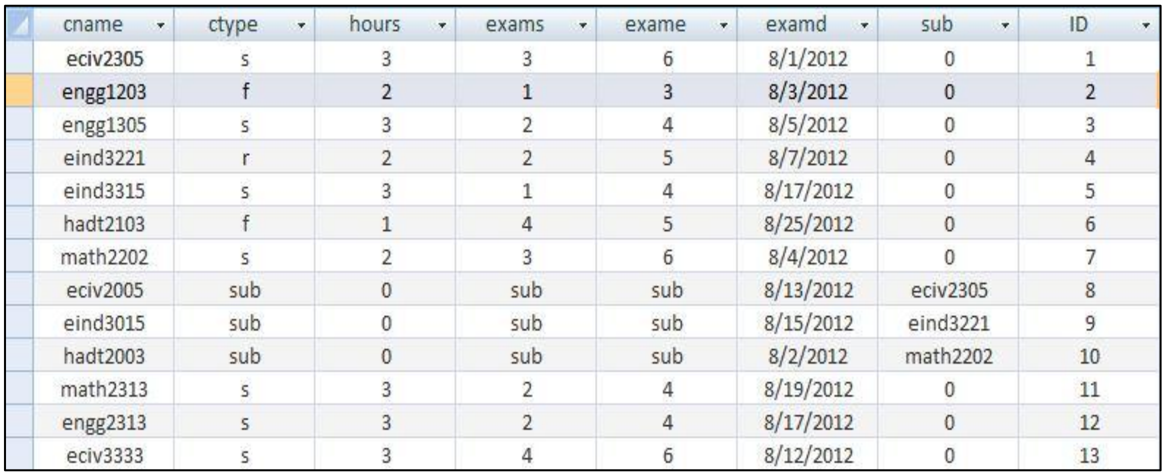
On the other hand, every single course in the available courses table should have an associated group of sections listed in the available sections table. These sections should meet only two conditions:

a. Relates to a course in the available courses list.
b. Offered for the current semester.
c. Not full yet.

Database Form

The database consists of two tables; the first table which is called "courses" (Figure 4.3) contains general information about the available courses which are:

A. Course name
B. Course number of credit hours.
C. Course type as a university, college or department requirement (with respect to the student major).
D. Final exam date.
E. Final exam starting time.
F. Final exam ending time.



cname	ctype	hours	exams	exame	examd	sub	ID
eciv2305	s	3	3	6	8/1/2012	0	1
engg1203	f	2	1	3	8/3/2012	0	2
engg1305	s	3	2	4	8/5/2012	0	3
eind3221	r	2	2	5	8/7/2012	0	4
eind3315	s	3	1	4	8/17/2012	0	5
hadt2103	f	1	4	5	8/25/2012	0	6
math2202	s	2	3	6	8/4/2012	0	7
eciv2005	sub	0	sub	sub	8/13/2012	eciv2305	8
eind3015	sub	0	sub	sub	8/15/2012	eind3221	9
hadt2003	sub	0	sub	sub	8/2/2012	math2202	10
math2313	s	3	2	4	8/19/2012	0	11
engg2313	s	3	2	4	8/17/2012	0	12
eciv3333	s	3	4	6	8/12/2012	0	13

Figure 4.3: Courses Table

The second table called "Sections" contains specific information about the available sections (Figure 4.4) which are:

A. Course name of the section.

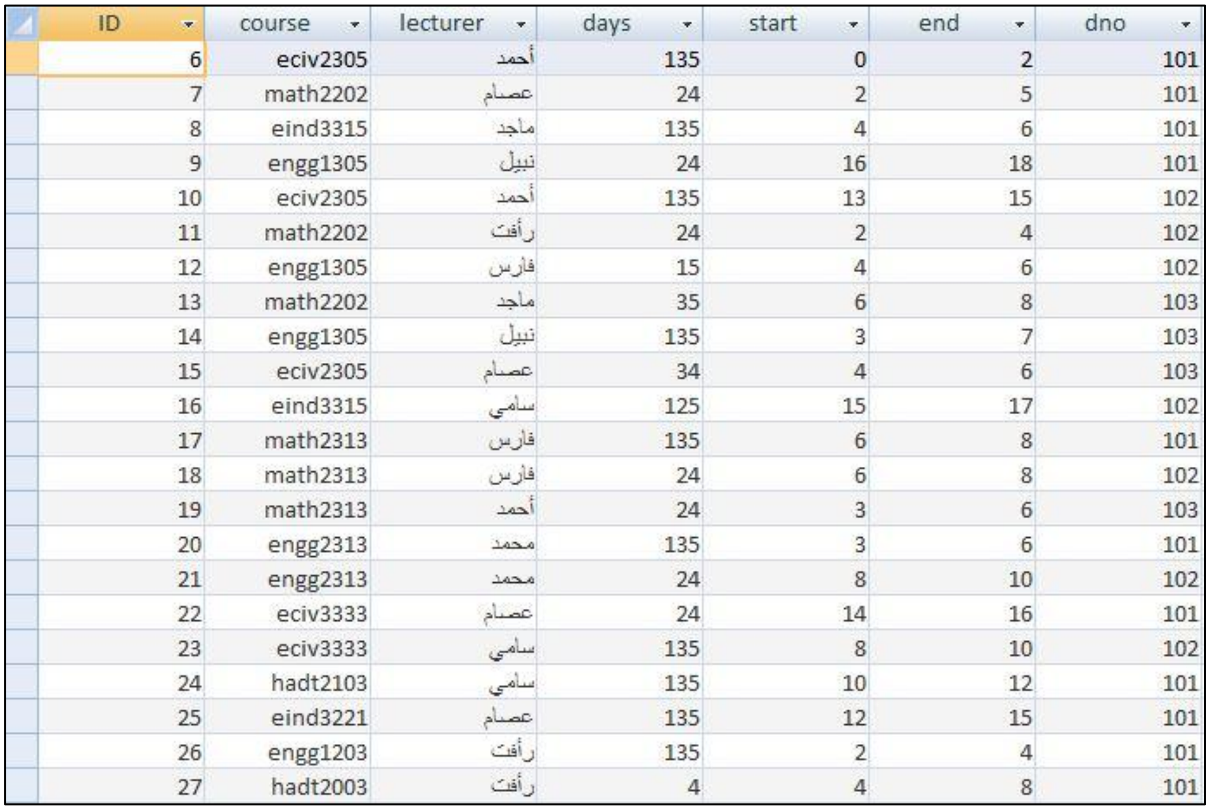
B. Section number.

C. Days in which sections are held.

D. Section starting time.

E. Section ending time.

F. Section lecturer name.



ID	course	lecturer	days	start	end	dno
6	eciv2305	أحمد	135	0	2	101
7	math2202	عصام	24	2	5	101
8	eind3315	ماجد	135	4	6	101
9	engg1305	نبيل	24	16	18	101
10	eciv2305	أحمد	135	13	15	102
11	math2202	رأفت	24	2	4	102
12	engg1305	فارس	15	4	6	102
13	math2202	ماجد	35	6	8	103
14	engg1305	نبيل	135	3	7	103
15	eciv2305	عصام	34	4	6	103
16	eind3315	سامي	125	15	17	102
17	math2313	فارس	135	6	8	101
18	math2313	فارس	24	6	8	102
19	math2313	أحمد	24	3	6	103
20	engg2313	محمد	135	3	6	101
21	engg2313	محمد	24	8	10	102
22	eciv3333	عصام	24	14	16	101
23	eciv3333	سامي	135	8	10	102
24	hadt2103	سامي	135	10	12	101
25	eind3221	عصام	135	12	15	101
26	engg1203	رأفت	135	2	4	101
27	hadt2003	رأفت	4	4	8	101

Figure 4.4: Sections Table

4.1.2 Model Formulation

As stated before, goal programming can be thought of as an extension or generalization of linear programming. Thus, it mainly depends on the mathematical representation of the problem in the form of an objective function that is subject to a set of constraints.

Goal programming problems can be categorized according to the type of mathematical programming model that it forms. The model produced by the system here can be called a regular zero-one integer goal programming. A non-preemptive goal programming problems refers to problems which goals are of comparable importance, while “zero–one Integer programming” refers to a special case of integer programming in which all the decision variables must have integer solution values of 0 or 1.

The basic approach of goal programming is to establish a specific numeric goal for each of the objectives, formulate an objective function for each objective, and then seek a solution that minimizes the (weighted) sum of deviations of these objective functions from their respective goals. There are three possible types of goals:

A. A lower, one–sided goal sets a lower limit that we do not want to fall under (but exceeding the limit is fine).
B. An upper, one–sided goal sets an upper limit that we do not want to exceed (but falling under the limit is fine).
C. A two–sided goal sets a specific target that we do not want to miss on either side [38].

The type of goal programming used in this system is the weighted one. It is also named regular GP. The weighted GP is used when the decision–maker is interested in direct comparisons of the objectives and is actually able to place a weight of importance for these objectives with respect to each other, which is usually the case in this scheduling issue. Besides, in this system, the user can chose not to consider all criteria.

Variables

A. Decision variables

The software is built based on the assumption that all information regarding the available courses in a certain semester are already assigned by the university such as the courses themselves, classes timing, final exams timing or lecturers. Whatever tools used to do that job have nothing to do with this software. This means that students have no choice but to choose whether to enroll in a certain course or not, if they choose to enroll it, then they will have to choose between its classes.

It may seem as if there are two kinds of decision variables:

A. Course decision variable: equals one in case the course is taken, 0 otherwise.
B. Sections decision variable: equals one in case a section is chosen, 0 otherwise.

However, in fact, it is just one, which is the section variable, since it carries the class specific information while implicitly means that the related course is chosen. Nevertheless, later we will see that a course binary variable is used to ease the formulation and the programmatic work.

A set of variables starting with the letter “d” will be declared to denote sections decision variables while another starting with the letter “c” is declared to denote courses decision variables.

The declaration of the courses variables will make it easier to formulate constraints that relate to course characteristic whether they are soft or hard.
Soft constraints that pertain to a course characteristic:

A. Desired number of credit hours in a semester.
B. Number of each Courses type.
C. Desired and undesired courses.
D. Empty Periods before final exams.

Hard constraints that pertain to a course characteristic:

A. Final exams timing conflicts.
B. Maximum and minimum allowed number of credit hours
C. The relation between courses and their subordinates.

Thus, instead of summing variables of all sections of all courses that are involved in a certain constraint, we can use the course variable only and then, a simple set of equations can be created to bind each group of sections to their associated course.

B. Deviation Variables

Most of the deviation variables that enter the objective function are binary variables, that is, it cannot be assigned values other than zero or one, however, there are some deviations that are set to be integers such as those used in the desired number of credit hours constraint, number of each course type and those used for the undesired lecturers constraints. Moreover, later it will be clear that it is not necessary to have both a positive and a negative deviation in all types of constraints.

On the other hand, there are some deviation variables that do not enter the objective function such as those which are only important to balance the equation.

The objective Function

As stated, the objective function will be to minimize the weighted sum of deviations of the various goals from their targeted values.

In this model, the objective function may turn out to be huge due to the large number of criteria which are 11. Each one of them may generate a large number of equations; each equation will contribute to the objective function with its positive deviation, negative deviation or both depending on the related criteria and the targeted goal. Besides, these deviations are named in a meaningful way for programmatic issues. These names may be formed of 2 to 6 characters. Structure of the objective function is shown in equation 4.1.

$$Min\, Z = \sum_{i=1}^{n} (\alpha_i\, d_i^+ + \beta_i\, d_i^-) \tag{4.1}$$

Where:

$d_i^+ \; and \; d_i^-$ respectively, over and under achievements of the ith goal.

α_i and β_i: are the respective positive weights attached to these deviations.

Assigning weights of the deviations in the objective function will be processed according to the results from the analytic hierarchy process incorporated in the software. These weights can be thought of as a determinant of the penalty placed upon being far from the desired goal. Distribution process of these weights will be discussed in the AHP part.

Soft Constraints

Represent the various goals viewed as equations which represent the criteria that matter to the student during registration. They are called soft because of the addition of both the positive and the negative deviations. This addition means that these equations cannot restrict the problem since the two deviations can be given any value, however, the optimization process will try to minimize as much as possible the value of those deviations whose weights in the objective function are high.

The structure of the equations built for each type of the various criteria will be discussed next.

A. Number of Courses in Each Type

which in turn consists of three parts:

A. The desired number of department requirements courses.

$$\sum_{i=1}^{s} C_s i + dns - dps = S \tag{4.2}$$

B. The desired number of faculty requirements courses.

$$\sum_{i=1}^{f} C_f i + dnf - dpf = F \tag{4.3}$$

C. The desired number of university requirements courses.

$$\sum_{i=1}^{r} C_r i + dnr - dpr = R \tag{4.4}$$

Where:

S,**F** and **R**: Denotes the desired amount of courses of each type.

S,**f**,**r**: Denotes the number available for registration of each type.

C_xi : Denotes the ith course of type x.

dnx, **dpx**: Denote the negative and the positive deviation of the equation of type x.

The user will also be able to determine ranges with respect to this amount as follows:

A. Greater than. only the negative deviation will be penalized.
B. Lower than. only the positive deviation will be penalized.
C. Equal to. both the negative and the positive deviations will be penalized.

B. The Desired Number of Credit Hours

$$\sum_{i=1}^{n} X_i \, . C_i + \; dhn - dhp = Dh \qquad (4.5)$$

Where:

Dh: Denotes the desired number of credit hours.

n: Denotes the total number of the available courses.

C_i : Denotes the ith course.

X_i: Denotes the number of credit hours of the ith course.

dhn, **dhp**: Denote the negative and the positive deviation of the equation.

Again, the user will be able to determine ranges for this number as greater, lower or equal to.

It should be noted that both deviation variables used in the desired number of each course type and the desired number of credit hours are integers –not binary–, since they can be assigned any value that is equal or greater than zero.

C. The Minimum Number Of Empty Days Between Exams.

$$C_a + C_b - Eapb + Eanb = 1 \quad (4.6)$$

Where:

C_a, C_b: denotes any couple of courses whose final exam dates separates by less than the desired period.

Eapb, **Eanb**: denotes both the positive and the negative deviations of the equation.

Note that only the positive deviation will be penalized since – as the equation states – giving this variable the value of one means that both courses variables whose final exam dates violate the desired separation period are also assigned the value of one. On the other hand, the negative deviation only exists to balance the equation in case both courses variables were assigned the value of zero.

The next step is to generate similar equations for all different couples of courses whose final exam dates violate the desired separation period. That would be a programming issue. Note that the same penalty will be assigned for all generated equations.

D. The Furthest Date Of Final Exams

Here comes the most confusing problem that faced the author during the formulation presses which is:

You cannot just simply sum courses variables which final exam date bypass the desired furthest date, making this sum equal to zero and then assign a penalty for the positive deviation, because this penalty will increase as the number of these variables whose values turned out to be one increases, however, the penalty of having one course or ten bypassing that date should be the same since the student goal failed anyway.

The same thing applies for both the desired empty periods and days throughout the week, because, for a student who needs a certain day to be entirely empty from lectures, it does not matter whether one lecture is assigned to that day or more because he will have to go anyway.

Multiplying the inverse of these variables, making them equal to one and then, assigning a penalty for the negative deviation will result in a nonlinear equation.

A solution to this problem was proposed as follows:

All decision variables that pertain to a course or a section whose characteristics violate the desired goal of such issue will be summed, then, a group of binary variables with the same number will be subtracted from this summation. Each variable will be multiplied by its order in the group. All of these subtracted binary variables will enter the objective function with the same weight, thus, the optimization process will give a value of only one to the binary variable that is multiplied by the number that represents the amount of decision variables who were assigned a value of one.

The generation process of such equations will of course be accomplished programmatically keeping in mind that the number of decision variables which should be included in each case is variable.

Thus, the constraint for the furthest date of final exams can be expressed as follows:

$$\sum_{i=1}^{n} Cv_i - \sum_{i=1}^{n} i \,.\, FED_i = 0 \tag{4.7}$$

Where:

C_{vi}: the ith course decision variable that violate the desired furthest date of final exams.

n: the number of the violating courses.

FED: a binary variable.

E. The Desired Courses

If the course decision variable was assigned a value of zero, then the negative binary deviation which is penalized in the objective function will be assigned a value of one as a penalty of not taking the desired course.

$$C_i + DC_i = 1 \tag{4.8}$$

Where:

C_i: The desired Course decision variable.

DC_i: a binary variable.

Note that there is no positive deviation because the course decision variable is binary, hence, it cannot take a value more than one.

The software interface allows for eight desired courses to be selected by the user, thus, the same equation will be generated for all courses selected.

F. The Preferred Lecturers

It is uncommon for a student to desire a certain lecturer regardless of the course. This is why the desired lecturer criteria is bound to the desired courses criteria, thus, a desired course should be selected first, then, a list with all of its available lecturers will be filled. The user will have the choice not to select a certain lecturer, as a result, the desired course constraint shown previously will be generated. But, in case the user selected a certain lecturer from the list, then the desired lecturer constraint shown in equation 4.9 will be generated instead, since it implicitly means that the associated course is also desired.

$$\sum_{i=1}^{n} d_i + dd_x = 1 \tag{4.9}$$

Where:

d_i: Denotes the ith section which course and lecturer are desired.

dd_x: a binary variable.

n: Denotes the number of sections course and lecturer are desired.

Again, there will be no need for a positive deviation. Note that the summation is made because there may be more than one section of that course that is given by that lecturer, however, assigning one section of them will be enough. Of course there is no way to assign more than one section of the same course, this will be the function of the hard constraints discussed later.

All eight slots available for the selection of the desired courses will have an associated eight lists that will be filled with the available lecturers of each course upon its selection. Thus, the previous constraint will be generated for all lecturers selected by the user.

G. The Desired Empty Days Before A Certain Course Final Exam

This criteria is also bound to the desired courses criteria. The user will be able to specify the minimum number of empty days that precedes the final exam date of every desired course he selects.

Constraints that represent this criteria is a bit more complicated. For one thing, it is meaningless to make one relationship between all courses whose final exams fall in the specified range, multiple relations should be constructed between the intended course and every violating course separately, for another, the same penalty should be placed for the violation of this period regardless the number of violations.
Thus, there will be two stages:

A. A set of bilateral relations is constructed between the intended course and all courses whose final exam fall in the desired empty period. All of these relations will include a binary variable which will be assigned the value of one if both courses were taken.

$$C_a + C_b - P_b p_a + P_b n_a = 1 \qquad (4.10)$$

Where:

C_a: Denotes the decision variable of the course which is assigned a period to precede its final exam date.

C_b: Denotes the decision variable of a course which final exam falls in the desired empty period assigned to precede the final exam of course “a”.

$P_b p_a$, **$P_b n_a$**: binary variables.

The same constraint will be constructed for all courses whose final exams fall in the desired empty period assigned to precede the final exam of course “a”.

B. A final constraint will be constructed between all binary variables included in the previous set of bilateral relations and the same number of another group of binary variables. This constraint ensures that the penalty will not be affected by the number of violations.

$$\sum_{i=1}^{n} P_{bi}p_a - i\,.\,C_aE_i \;=\; 0 \qquad (4.11)$$

Where:

$P_{bi}p_a$: a binary variable which denotes that course “b” violates the desired period of course “a’.

C_aE_i: a binary variable that goes in the objective function.

n: the number of courses whose final exams fall in the desired empty period before course “a”.

The previously mentioned two stages will be performed for all courses which are assigned a period to precede its final exam date.

H. The Undesired Courses

One would think, if there are a group of courses that are undesirable to the student, then, why would not they be excluded from the backend database in the first place?. This necessitate a clear definition of the term “undesired courses”. Undesired courses refer to available courses that –for some reason– a student is trying to avoid, however, he/she accepts to enroll in these courses as a final resort lest he/she should fill in the trap of an infeasible solutions due to unfulfilled hard constraint of the minimum number of credit hours that has to be registered per semester, or because there are other criteria that are more important to the student that necessitate acceptance of these courses, such as a desired amount of credit hours or a desired amount of a certain courses type.

These courses being excluded from the backend database means that they will not enter the optimization process in the first place, as a result, these courses will not appear what so ever in the solution.

$$C_i - UC_i = 0 \tag{4.12}$$

Where:

C_i: Denotes the decision variable of the undesired course.

UC_i: a binary variable.

If the course decision variable was assigned a value of one, then the positive binary deviation which is penalized in the objective function will also be assigned a value of one as a penalty of taking the undesired course.

The software interface allows for eight undesired courses to be selected by the user, thus, the same equation will be generated for all courses selected.

I. The Undesirable Lecturers

$$\sum_{i=1}^{n} d_{di} - UL_x = 0 \tag{4.13}$$

Where:

d_{di}: Denotes the ith section given by the xth undesired lecturer.

UL_x: an integer variable.

Note that this constraint is built to include all sections given by the undesired lecturer regardless of the course. Because of that, the deviation variable here is integer because more than one section that are given by the undesired lecturer may be assigned, consequently, the deviation variable will increase which in turn will cause the penalty to increase.

The software interface allows for eight undesired lecturers to be selected by the user, thus, the same equation will be generated for all lecturers selected. The selection will be made from a list that is filled with all lecturers of the available sections once the program operates.

J. The Desired empty days throughout the week

As mentioned in the furthest date of final exams criteria, the most important point here is to unify the penalty of not being able to empty a day entirely with respect to the number of violations (lectures assigned to that day). Thus:

$$\sum_{i=1}^{n} d_{yi} - i\,.ED_{yi} = 0 \tag{4.14}$$

Where:

d_{yi}: Denotes the ith section that meets on the yth day.

ED_{yi}: a binary variable.

n: Denotes the number of sections held on the yth day.

The same constraint will be constructed for all days desired to be empty.

K. The Desired Empty Periods Throughout the week

A desired empty period on a certain day refers to the interval of time throughout that day at which a student prefers not to appoint lectures.

It should be noted that a student will still has the choice of defining a desired empty period within a certain day even though that day was set as a desired empty day in the ninth criteria. It may seem as if it is going to cause double penalty, however it will not.

Empting a certain day is a thing, empting a certain period within it is another. Failing to empty a certain day does not necessarily means failing to avoid a certain period within it. A student may wish to define a desired empty period within a day that is set to be empty. It is a way of guiding the optimization process to avoid this particular period just in case it had no other choice but to assign courses to that day.

On the other hand, double penalty will apply only if the defined desired empty period within a certain day was large enough to involve all possible sections in that day.

$$\sum_{i=1}^{n} d_{yi} - i \,.\, EP_{yi} = 0 \tag{4.15}$$

Where:

d_{yi}: Denotes the ith section that meets in the undesired period of the yth day.

EP_{yi}: a binary variable.

n: Denotes the number of sections held in the undesired period of the yth day.

The same constraint will be constructed for all days that include an undesired period.

Usually lectures begin and end at an exact hour during the day, however, sometimes, they may start or end at half an hour. Nevertheless, and for the sake of thoroughness, time was incorporated into this system in term of units such that each unit represents half an hour.

Note that what is actually meant by sections that meet in the desired empty period is all of those sections which timing intersects with that period by at least the smallest amount of time which is in this case “half an hour”.

Timing conflicts

To detect conflict in timing, five cases where defined as the only states at which two periods are said to be intersecting each other. Suppose we have a primary period known as “P” and a secondary one known as “S”. Ps, Pe, Ss, Se denote the starting time of the primary period, the ending time of the primary period, the starting time of the secondary period and the ending time of the secondary period respectively.

These cases – also viewed in figure 4.5 – are as follow:

A. (Ss < Ps) And (S > (Ps – Ss)).
B. (Se > Pe) And (S > (Se – Pe)).
C. (Ps < Ss) And (Pe > Se).
D. Ps = Ss.
E. Pe = Se.

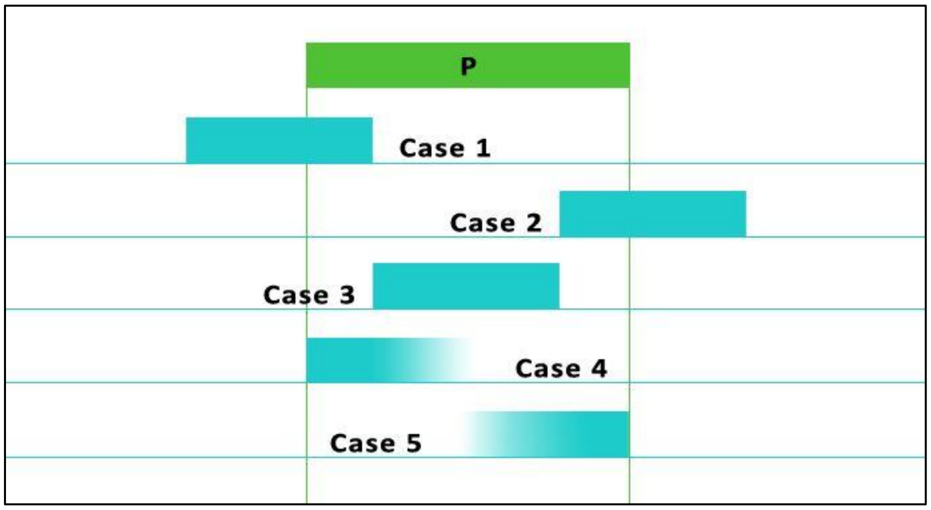


Figure 4.5: Cases of Conflicting Periods of Time

The same principle was also applied to check for both lectures and final exams timing conflicts in the hard constraints section coming next.

Hard Constraints

Represent laws of the academic registration system, some axiomatic issues and some relations which importance arise due to the way by which model variables are defined.

A. Maximum and Minimum Allowed Number of Credit Hours

By default, the maximum number is set to 22 and the minimum is set to 12, which are the limits in effect in the Islamic university of Gaza for the regular student in any faculty. However, the software grants the user the ability to change these values, because in some certain situations, the student will be allowed to break these limits (graduate students for example).

a. The maximum number of credit hours

$$\sum_{i=1}^{n} X_i \, . C_i \ \leq MAXH \tag{4.16}$$

b. The minimum number of credit hours

$$\sum_{i=1}^{n} X_i \, . C_i \ \geq MINH \tag{4.17}$$

Where:

X_i: Denotes the number of credit hours of the ith course.

C_i: Denotes the decision variable of the ith course.

n: Denotes the number of all available courses.

MAXH: Denotes the maximum allowed number of credit hours

MINH: Denotes the minimum allowed number of credit hours

B. Final Exams Conflicts

It means those courses which final exams timing overlaps. A bilateral set of hard constraints will be constructed between any pair of courses which final exams timing overlaps as follows:

$$C_a + C_b \leq 1 \tag{4.18}$$

Where:

C_a, C_b: Denotes the decision variables of any pair of courses which final exams timing overlaps.

The previous equation will allow at most for one course of this pair to be assigned.

C. Courses And their Related Sections

As stated before, the declaration of the courses decision variables will make it easier to formulate constraints that relate to course characteristic – although they are not really needed – it also makes it easier to write the generation code of such constraints, however, it is necessary to bind each group of sections to their related course. This is because the final output of the software will be presented in sections not in courses. Thus, these set of constraints will be responsible for binding all constraints – soft or hard – which contain courses decision variables to the model.

$$C_1 - \sum_{i=1}^{x1} d_{1i} = 0$$

$$\vdots$$

$$C_m - \sum_{i=1}^{xm} d_{mi} = 0 \tag{4.19}$$

Where:

$\mathbf{C_1}$: Denotes the decision variable of the first available course.

$\mathbf{X_1}$: Denotes the number of sections related to the first course.

$\mathbf{d_{1i}}$: Denotes the decision variable of the ith section that relates to the first course.

m: Denotes the number of available courses.

These constraints will play another important role by preventing selection of more than one section of the same course, since the summation of each section group is equal to a binary variable.

D. Lectures Conflicts

Represent constraints that prevent lectures timing conflicts. A bilateral set of hard constraints will be constructed between any pair of sections which any of its meetings timing overlaps with the other.

$$d_a + d_b \leq 1 \tag{4.20}$$

Where:

d_a, d_b: Denotes the decision variables of any pair of sections which any of its meetings timing overlaps with the other.

The previous constraint will allow at most for one section of this pair to be assigned.

E. Courses And Their Subordinates

The function of this type of constraints is to bind each course to its subordinate classes. These classes should be registered along with their related courses in the same semester, such as discussions or laboratory classes. The constraint shown in equation 4.21 will bind these subordinate classes in such a way that if a course is taken then its subordinate class will be taken as well. These subordinates will be treated by the system as normal courses Because, they also will have their own sections each of which will have its own timing and lecturer. Thus, those classes will be included in the courses table the same way as other normal courses, however, usually, they will have no credit hours, nor final exam information.

These subordinates are recognized by the software by their names. It is their names that enable the software bind them to their related courses.

$$C_a - C_b = 0 \tag{4.21}$$

Where:

C_a: Denotes the decision variable of a course “a”.

C_b: Denotes the decision variable of a course “b” which is a subordinate for course “a”.

4.2 The Analytic Hierarchy Process (AHP)

The previously discussed ten criteria are not always of the same importance to the student, besides, the student may not consider them all. A student may chose a course as a desired, a lecturer as undesired, a minimum number of credit hours and in the same time a certain day to be empty, however, the last goal is the one that matters to him most, so, there must be a way to enable the user to represent his priorities since most of the previously mentioned goals will be conflicting within the narrow framework of the available sections and the hard constraints during the optimization process. That is why the analytic hierarchy process was incorporated in the system, so that it can be used to derive weights or priorities that will be assigned to the various goals involved in the model. These weights will be derived from a set of pairwise comparisons established between the goals involved.

Other reasons as why AHP is used:

A. Humans are much more capable of making relative rather than absolute judgments.
B. AHP incorporates redundancy through pairwise comparisons, which results in a reduction of measurement error [39].

Decomposition is the first phase of AHP in which this problem is structured into two levels. The first level is represented by the main cluster which includes the ten main criteria which are:

A. The desired number of credit hours.
B. The desired number of each course type.
C. Minimum empty days between final exams.
D. The furthest date of final exams.
E. The desired courses. (Within available) or the desired courses given by the desired lecturers.
F. Number of empty days before a certain course final exam.
G. The undesired courses.
H. The undesired lecturers.
I. The desired empty days in schedule.
J. The desired empty periods in schedule.

Some criteria are ramified into a set of elements, which give the user the choice to determine a set of sub-objectives that follow the same main title. Such as:

A. The desired number of each course type.
B. The desired courses. (Within available) or the desired courses given by the desired lecturers.
C. Number of empty days before a certain course final exam.
D. The undesired courses.
E. The undesired lecturers.
F. The desired empty days in schedule.
G. The desired empty periods in schedule.

These sub-objectives will be treated into a group of sub-clusters which represent the second level. Figure 4.6 shows hierarchal structure of objectives.

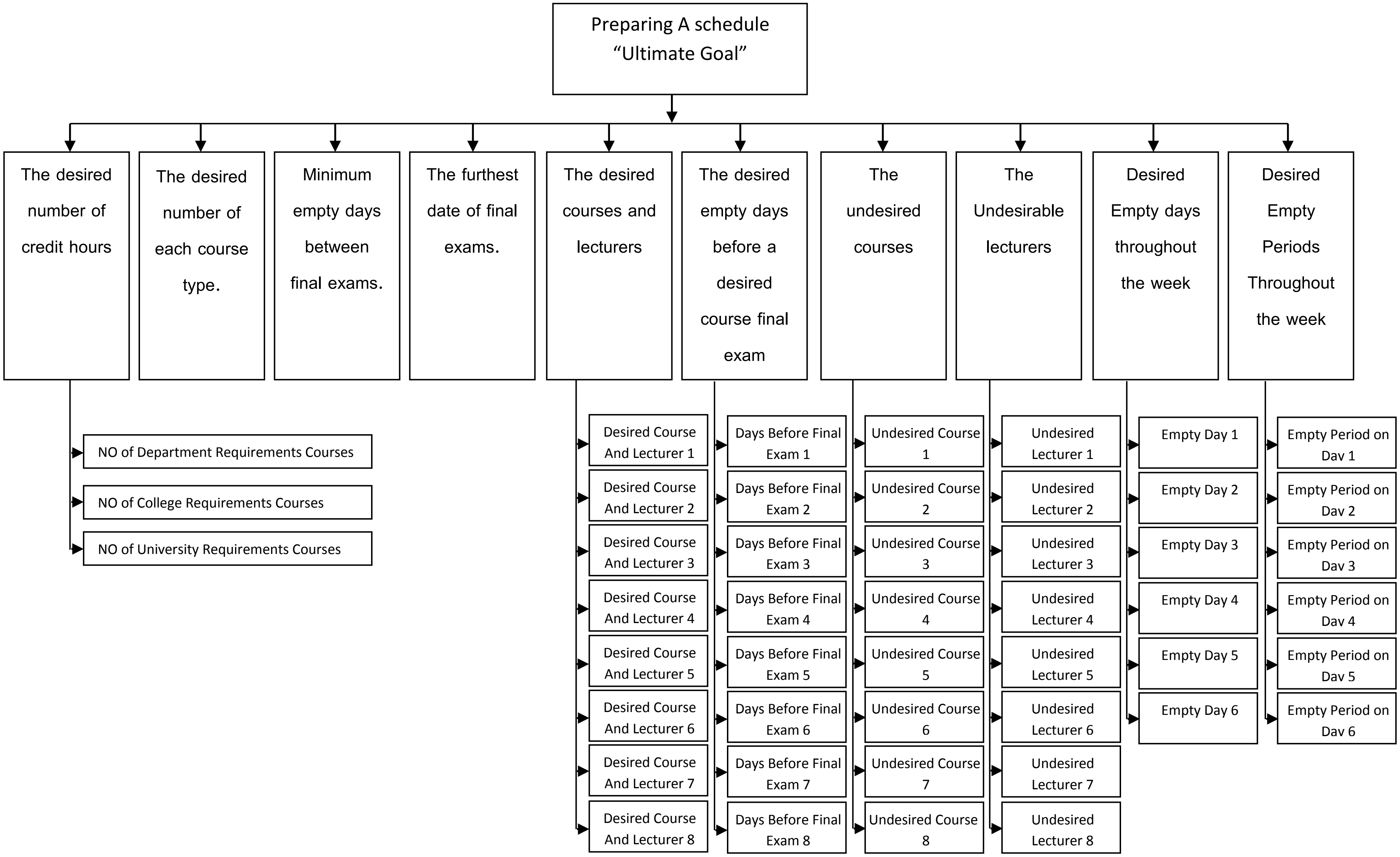

Preparing A schedule "Ultimate Goal"
The desired number of credit hours
The desired number of each course type.
Minimum empty days between final exams.
The furthest date of final exams.
The desired courses and lecturers
The desired empty days before a desired course final exam
The undesired courses
The Undesirable lecturers
Desired Empty days throughout the week
Desired Empty Periods Throughout the week
NO of Department Requirements Courses
NO of College Requirements Courses
NO of University Requirements Courses
Desired Course And Lecturer 1
Desired Course And Lecturer 2
Desired Course And Lecturer 3
Desired Course And Lecturer 4
Desired Course And Lecturer 5
Desired Course And Lecturer 6
Desired Course And Lecturer 7
Desired Course And Lecturer 8
Days Before Final Exam 1
Days Before Final Exam 2
Days Before Final Exam 3
Days Before Final Exam 4
Days Before Final Exam 5
Days Before Final Exam 6
Days Before Final Exam 7
Days Before Final Exam 8
Undesired Course 1
Undesired Course 2
Undesired Course 3
Undesired Course 4
Undesired Course 5
Undesired Course 6
Undesired Course 7
Undesired Course 8
Undesired Lecturer 1
Undesired Lecturer 2
Undesired Lecturer 3
Undesired Lecturer 4
Undesired Lecturer 5
Undesired Lecturer 6
Undesired Lecturer 7
Undesired Lecturer 8
Empty Day 1
Empty Day 2
Empty Day 3
Empty Day 4
Empty Day 5
Empty Day 6
Empty Period on Dav 1
Empty Period on Dav 2
Empty Period on Dav 3
Empty Period on Dav 4
Empty Period on Dav 5
Empty Period on Dav 6

Figure 4.6: Hierarchal Structure of Objectives

4.2.1 Application of AHP

It should be noted that the first criteria allows for three sub-objectives. Criteria from 5 to 8 allow for 8 sub-objectives representing the eight slots available for the selection of the desired and the undesired courses and lecturers, in addition to the eight slots available for the determination of the desired empty days before some certain courses final exams.

The last two criteria allow for 6 sub-objectives representing the desired empty days or periods throughout the 6 days of the weak.

However, only those objectives –elements- set by the user will be considered, whether at the first or the second level.

The second step is to construct pairwise comparisons of all combinations of elements in all clusters. The importance of any element can be compared to any other element in the same cluster on a scale that ranges from 1/9 to 9 using a comparison matrix which dimension depends on the included elements. Yet, elements of the same cluster will be given equal importance in case their comparison matrix was not manipulated.

Only half of the matrix will be visible to the user on the form of a triangle such that the intersection of a certain element row and another's column represents the relative value of the first with respect to the last. Thus, the other matrix half will be automatically calculated as the inverse of the first one.

To calculate priorities of the various elements in a certain pairwise comparisons matrix, AHP uses the Eigen value method to compute the priority vector which is the normalized Eigen vector of the matrix. However, an approximation of the Eigen value method is usually used in AHP due to its ease and precision.

Steps to compute the Eigen vector are as follow:

A. Sum each column of the reciprocal matrix.
B. Divide each element of the matrix with the sum of its column to get a normalized relative weight.
C. The normalized Eigen vector can be obtained by averaging through the rows.

The third step of AHP is to apply the principle of hierarchic composition by multiplying the local priorities of elements in a cluster by the 'global' priority of the parent element, producing global priorities or final scores throughout the hierarchy.
Usually the final step of AHP is to evaluate the existing alternative as to how much they would contribute to the achievement of the ultimate goal. This is done by adding up the importance of the lowest level elements that each alternative satisfies. However in this problem, there are no alternatives. Final scores will be used as weights in the objective function of the integer goal programming model are solved to yield the optimum alternatives.
Final weight of each considered element will be distributed among all of its deviation variables that go into the objective function (discussed in the formulation part).

Use of the analytic hierarchy process is not just the most important step, it is the only step that determines the quality of the software output. The slightest change in the objective function weights could result in a completely different schedule. Hence, it is very important for the user to be precise during this process.

The most important thing for any user is that he should know what he is comparing, particularly in the main level. The ten main criteria are clearly dissimilar, nevertheless, at least the user should know the units of each criterion that are panelized at the corresponding weight of that criteria, thus, he/she will be able to compare the importance of these units with respect to each other in the pairwise comparison process of the main matrix.

4.2.2 Criteria penalization:

Table 4.1 defines the units at which each of the ten criteria is penalized for being far from the corresponding specified goal.

Table 4.1: Criteria Penalization

NO	Criteria	Penalized Unit
1	The desired number of credit hours.	Each single hour outside the desired range specified by the user
2	The desired number of each course type.	Each single course outside the desired range of each type.
3	Minimum empty days between final exams.	One violation of this period (one case at which two courses final exams separate by less than the desired period).
4	The furthest date of final exams.	To violate this date (no matter how many exams bypass that date)

Table 4.1: Criteria Penalization (Continued).

5	The desired courses. (Within available) or the desired courses given by the desired lecturers.	Not to take a desired course or not to take a desired course with the desired lecturer in case one is specified.
6	Number of empty days before a certain course final exam.	To violate this period (some other courses final exams fall in this period no matter how many).
7	The undesired courses.	To take one undesired course.
8	The undesired lecturers.	To take one course that is given by un undesired lecturer.
9	The desired empty days in schedule.	Not to entirely empty a certain day that is set to be empty.
10	The desired empty periods in schedule.	Not to entirely empty a certain period in a certain day that is set to be empty.

4.3 The Software

This section focuses on the applied part of the system represented by the computer software itself. A general view of the software will be presented starting with the criteria part and going through the AHP part, model control, solving and the results. Finally guidance to the proper use of the software is given.

4.3.1 A background of the software developing tools.

The software was developed using visual basic 6; its related database is designed on Microsoft access 2007 and used Microsoft's ActiveX Data Objects (ADO) to access data stores. Also, the Structured Query Language (SQL) was used to query tables.

Visual Basic is a third-generation event-driven programming language from Microsoft first released in 1991. Visual Basic is designed to be relatively easy to learn and use. Visual Basic was derived from BASIC and enables the rapid application development (RAD) of Graphical User Interface (GUI) applications, access to databases using Data Access Objects, Remote Data Objects, or ActiveX Data Objects, and creation of ActiveX controls and objects. The final release was version 6 in 1998.

Like the BASIC programming language, Visual Basic was designed to be easily learned and used by beginner programmers. The language not only allows programmers to create simple GUI applications, but also to develop complex applications. Programming in VB is a combination of visually arranging components or controls on a form, specifying attributes and actions of those components, and writing additional lines of code for more functionality. Since default attributes and actions are defined for the components, a simple program can be created without the programmer having to write many lines of code. Performance problems were experienced by earlier versions, but with faster computers and native code compilation this has become less of an issue.

Although VB programs can be compiled into native code executables from version 5 onwards, they still require the presence of runtime libraries of approximately 1 MB in size. Runtime libraries are included by default in Windows 2000 and later, however for earlier versions of Windows, i.e. 95/98/NT, runtime libraries must be distributed together with the executable.

Forms are created using drag-and-drop techniques. A tool is used to place controls (e.g., text boxes, buttons, etc.) on the form (window). Controls have attributes and event handlers associated with them. Default values are provided when the control is created, but may be changed by the programmer. Many attribute values can be modified during run time based on user actions or changes in the environment, providing a dynamic application. For example, code can be inserted into the form resize event handler to reposition a control so that it remains centered on the form, expands to fill up the form, etc. By inserting code into the event handler for a key press in a text box, the program can automatically translate the case of the text being entered, or even prevent certain characters from being inserted.

Visual Basic can create executables (EXE files), ActiveX controls, or DLL files, but is primarily used to develop Windows applications and to interface database systems. Dialog boxes with less functionality can be used to provide pop-up capabilities. Controls provide the basic functionality of the application, while programmers can insert additional logic within the appropriate event handlers. For example, a drop-down combination box will automatically display its list and allow the user to select any element. An event handler is called when an item is selected, which can then execute additional code created by the programmer to perform some action based on which element was selected, such as populating a related list.

Microsoft Access, also known as Microsoft Office Access, is a database management system from Microsoft that combines the relational Microsoft Jet

Database Engine with a graphical user interface and software-development tools. It is a member of the Microsoft Office suite of applications

In addition to using its own database storage file, Microsoft Access may also be used as the 'front-end' with other products as the 'back-end' tables, such as Microsoft SQL Server and non-Microsoft products such as Oracle and Sybase. Multiple backend sources can be used by a Microsoft Access Jet Database (accdb and mdb formats). Similarly, some applications will only use the Microsoft Access tables and use another product as a front-end, such as Visual Basic or ASP.NET. which is done in this project.

Microsoft Data Access Components (commonly abbreviated MDAC; also known as Windows DAC) is a framework of interrelated Microsoft technologies that allows programmers a uniform and comprehensive way of developing applications that can access almost any data store. Its components include: ActiveX Data Objects (ADO), OLE DB, and Open Database Connectivity (ODBC), figure 4.7 shows these components.

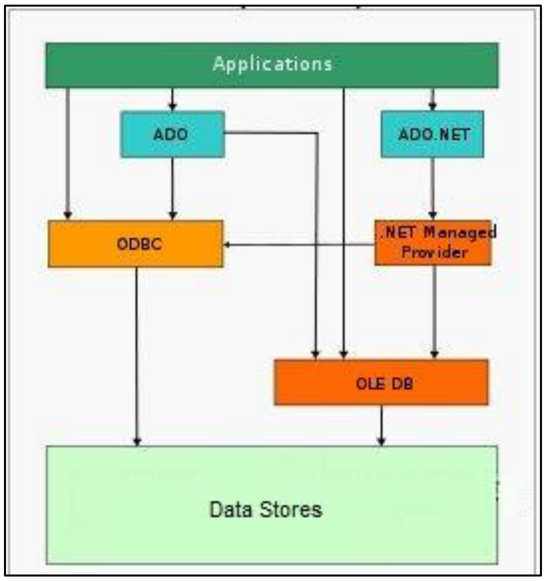


Figure 4.7: MDAC (Microsoft Data Access Components)

Microsoft's ActiveX Data Objects (ADO) is a set of Component Object Model (COM) objects for accessing data sources. A part of MDAC, it provides a middleware layer between programming languages and OLE DB (a means of accessing data stores, whether they be databases or otherwise, in a uniform manner). ADO allows a developer to write programs that access data without knowing how the database is implemented; developers must be aware of the database for connection only. No knowledge of the Structured Query Language (SQL) is required to access a database when using ADO, although one can use ADO to execute SQL commands directly.

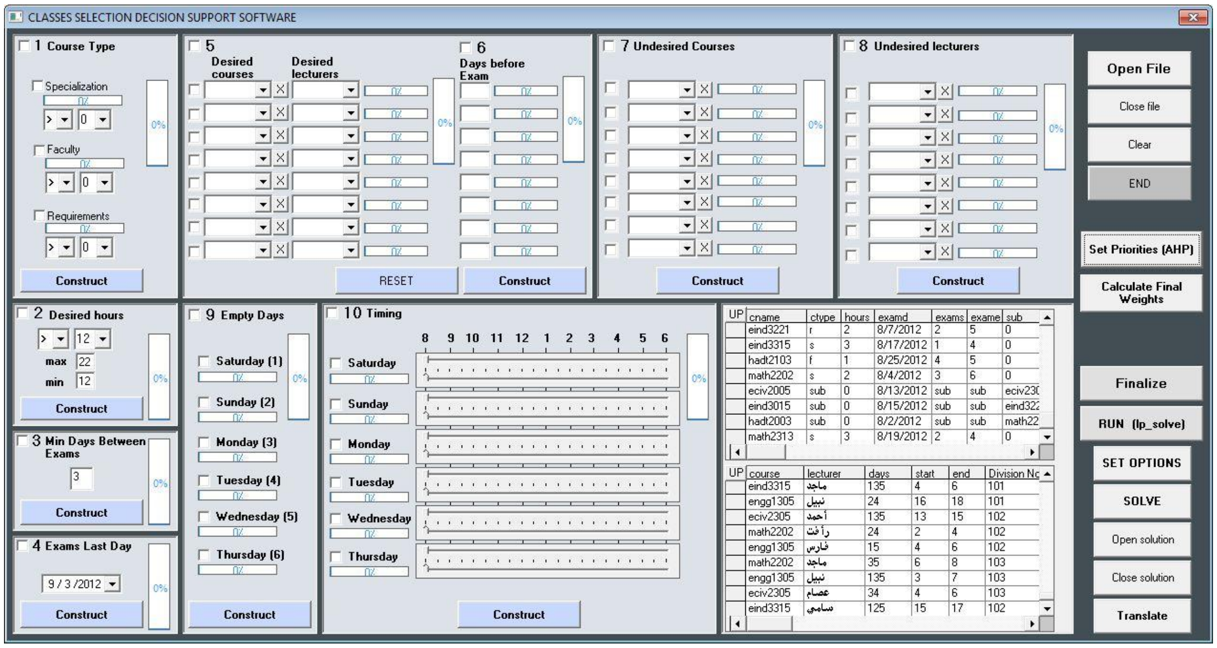
Structured Query Language (SQL) is a special-purpose programming language designed for managing data in relational database management systems (RDBMS). SQL was first created by Edgar F. Codd, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks". It became the most widely used database language. Also, SQL became a standard of the American National Standards Institute (ANSI) in 1986.

4.3.2 The Interface

Today the graphical user interfaces (GUIs) is in a growing importance. Besides, a software interface is the most visible component of an OBDSS. However, an attractive user interface is not considered so far since the project is yet to be built. Also, it is very difficult for one person to develop the fundamental design of a model, write the code of a large and complicated computer software, design a query system that binds the software with the university database and in the same time care about a fancy interface, since all of the previously mentioned tasks was carried out by the author alone.

Right now, the interface is clearly stuffed with too many parts and objects – as shown in figure 4.8 –. This is done essentially for the sake of a clear and explicit presentation. Nevertheless, a friendlier interface may be developed later.

As for now, the software interface is divided into three sections. The first section is occupied by the main ten criteria. This section is for the user to determine his goals regarding the criteria that concern him. The second section contains a view of the software related database including both courses and sections tables. The third section is for control buttons.



2 Desired hours: > 12, max 22, min 12

3 Min Days Between Exams: 3

4 Exams Last Day: 9 / 3 /2012

UP	cname	ctype	hours	examd	exams	exame	sub
	eind3221	r	2	8/7/2012	2	5	0
	eind3315	s	3	8/17/2012	1	4	0
	hadt2103	f	1	8/25/2012	4	5	0
	math2202	s	2	8/4/2012	3	6	0
	eciv2005	sub	0	8/13/2012	sub	sub	eciv230
	eind3015	sub	0	8/15/2012	sub	sub	eind322
	hadt2003	sub	0	8/2/2012	sub	sub	math22
	math2313	s	3	8/19/2012	2	4	0

UP	course	lecturer	days	start	end	Division No
	eind3315	ماجد	135	4	6	101
	engg1305	نبيل	24	16	18	101
	eciv2305	أحمد	135	13	15	102
	math2202	رأفت	24	2	4	102
	engg1305	فارس	15	4	6	102
	math2202	ماجد	35	6	8	103
	engg1305	نبيل	135	3	7	103
	eciv2305	عصام	34	4	6	103
	eind3315	سامي	125	15	17	102

Figure 4.8: Classes Selection Decision Support Software Interface

A. Criteria Section

a. Course Type

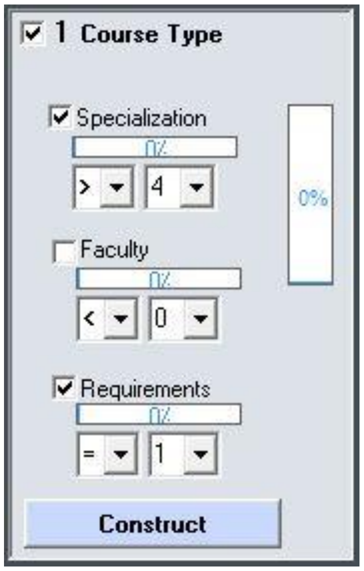


Figure 4.9: Course Type

As shown in (figure 4.9) this criteria contains three parts, departmental, faculty and university requirements. The user will be able to set whatever parts he wants or leave them all. To define a goal, the user should set two combo-boxes, one for the desired number of courses and the other is for desired range whether it is greater than, less than or equal to. The combo box for the desired number of courses is filled upon student login with numbers from 0 to the maximum number of courses that exist of the associated type in the database. The button named “construct” is responsible of constructing the required constraints in the model document according to what is set by the user.

b. The Desired Number Of Credit Hours

Again, the user will have to set two combo boxes, one is for the range operator and the other is for the desired number of credit hours as shown in figure 4.10. Also, the maximum and the minimum allowed number of credit hours are set in this box. Although these two values relate to hard constraints, they are the determinants of the desired number of hour's combo box list of values. Of course it is illogical to desire a number of credit hours outside this list.

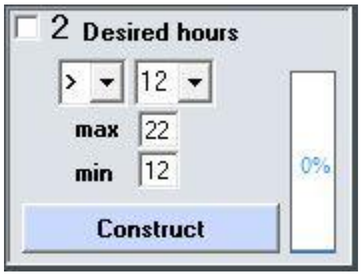


Figure 4.10: Desired Hours

c. Minimum number of days between final exams

Here, the user will just have to specify the minimum number of days he wishes to separate between final exams inside the only text box there (figure 4.11). This text box which only accepts numbers can be filled by only one number.

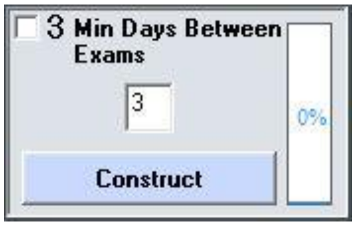


Figure 4.11: Minimum Number of Days between Final Exams

d. Final Exams Last day

The user will not have to enter the date manually, once this date picker box is clicked, a calendar will show up (figure 4.12) to choose from.

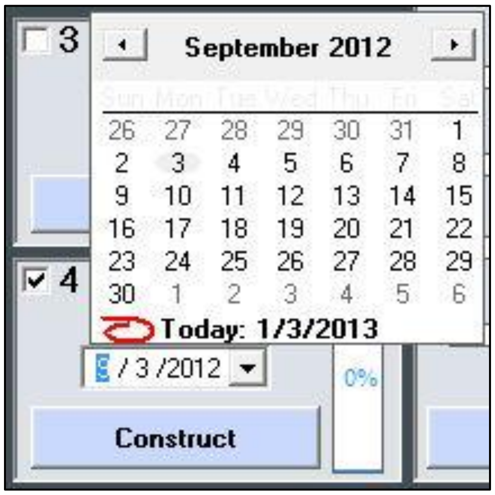


Figure 4.12: Final Exams Last day

e. The Desired Courses, The Desired Lecturers And The Desired Period Before a Certain Course

Both the desired lecturer and the desired period before a certain course final exam are bound to the desired courses. As stated before, it is uncommon for a student to desire a certain lecturer regardless of the course. Although it is not justified to bind the desired period before a certain course final exam to the desired courses, these two criteria will only be allowed to be set once a desired course is selected.

As in (Figure 4.13), there are eight combo boxes for the desired courses criteria. These boxes are filled upon user login with all available courses. Since the number of credit hours per course is usually three, eight courses will be more than enough, however, the idea behind the desired courses criteria has nothing to do with the study load, and it is about giving the software more alternatives. Once a course is selected in any of the desired courses criteria combo boxes, it will be subtracted from all other combo boxes list items, moreover, the associated combo box of the desired lecturer combo box will be filled by all lecturers who give this course. Besides, the associated text box of the desired period before final exam will be enabled.

The reset button will empty all boxes and refill the desired courses combo boxes with the available courses again.

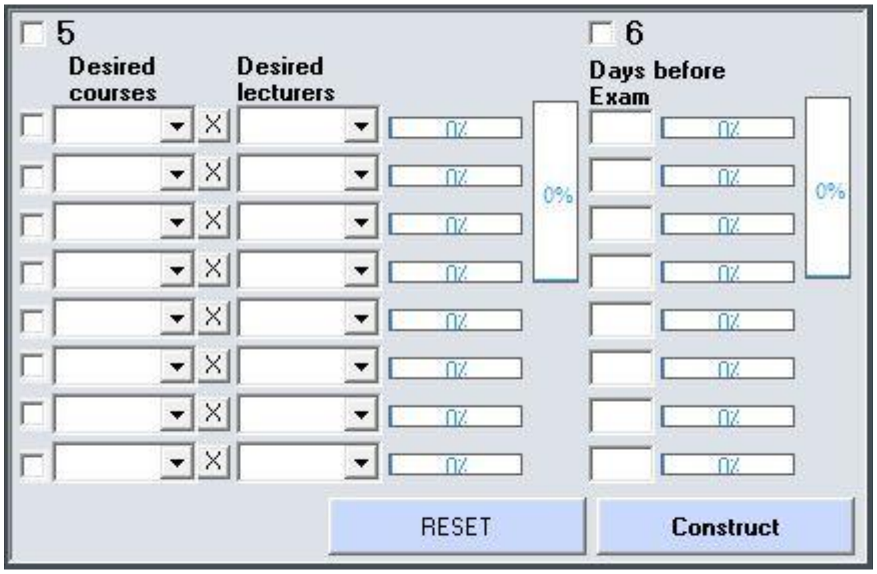


Figure 4.13: The Desired Courses, the Desired Lecturers And The Desired Period before a Certain Course

f. The Undesired Courses

This section is composed of eight combo boxes that are filled with all available courses upon user login the same way as in the desired courses section (figure

4.14). In both the desired and the undesired criteria sections, user can select courses from any scattered group of combo boxes, they should not necessarily be successive.

Also, the desired and the undesired courses sections are bound in a way such that any course selected in any section will be subtracted from the other section combo boxes list items. This is done to prevent selecting the same course in both sections since it does not make sense.

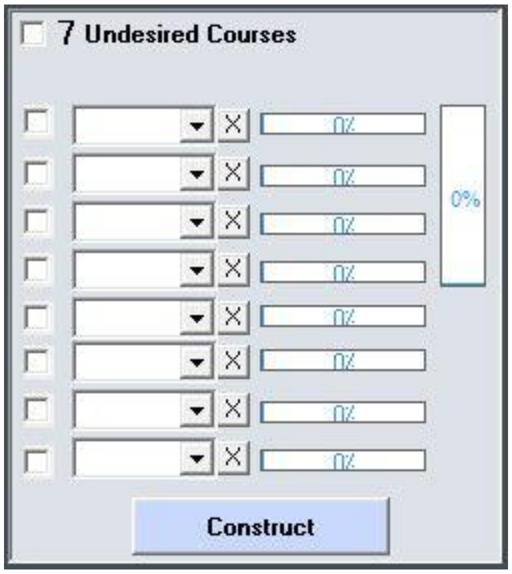


Figure 4.14: The Undesired Courses

g. The Undesired Lecturers

This section is also composed of eight combo boxes that are filled upon user login with all lecturers of all available sections (figure 4.15). This section is bound to the desired lecturers section. Whenever a lecturer is selected in any section, he will be subtracted from all of the other section combo boxes list items. Also, whenever a lecturer is deleted in any section, he will be relisted in all of the other section combo boxes. A gain these combo boxes can be chosen randomly.

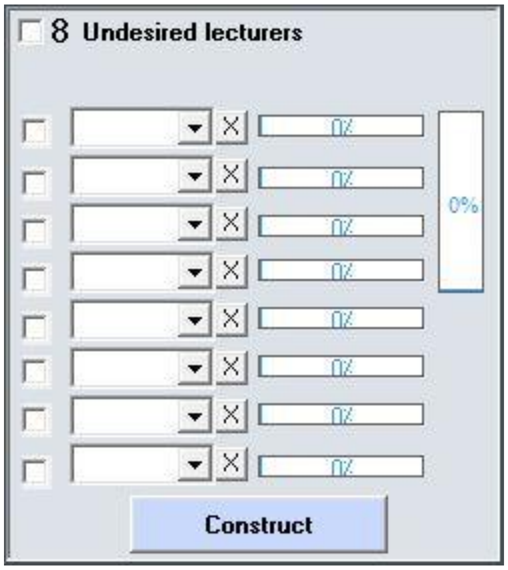


Figure 4.15: The Undesired Lecturers

h. The Desired Empty Days.

This section consists of six checkboxes representing days from Saturday to Thursday respectively (Figure 4.16). The user needs just to check those boxes of the days which he desires to be empty.

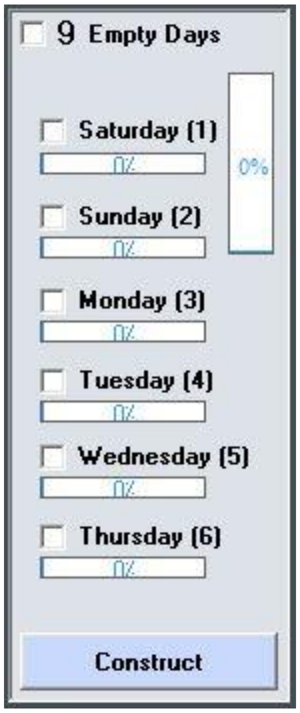


Figure 4.16: The Desired Empty Days

i. The Desired Empty Periods

This section consists of six boxes representing days from Saturday to Thursday. Inside each box there are two slider tools gradual into 20 steps such that 0 represents eight o'clock morning while 20 represents six o'clock afternoon so each step represents half an hour (Figure 4.17).

To specify an empty period in a certain day, user should set the two slider tools of that day to the timings at which that period starts and ends. It does not matter which slider tool represents which timing.

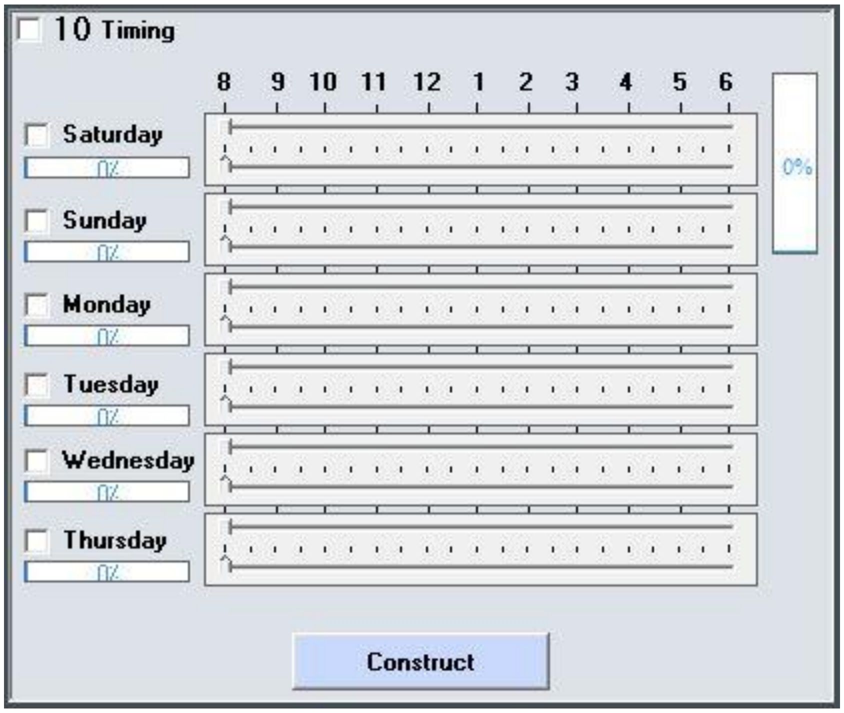


Figure 4.17: The Desired Empty Periods

4.3.3 The Analytic Hierarchy Process (AHP) Part

AHP can be accessed through the button named "Set Priorities (AHP)" which leads to the form shown in figure 4.18. This form navigates through the various AHP clusters. Each button in this form lead to a form that contains a group of combo boxes with the shape of a half matrix. This half matrix provides a way by which users can assign pairwise comparisons between the various concerned elements.
The button named "main" leads a form that establish a pairwise comparison between the main ten criteria while the rest of the buttons lead to forms that establish a pairwise comparison between sub-elements related to any of the ramified criteria. The number in these buttons names refer to the number of the associated main criteria in the interface.

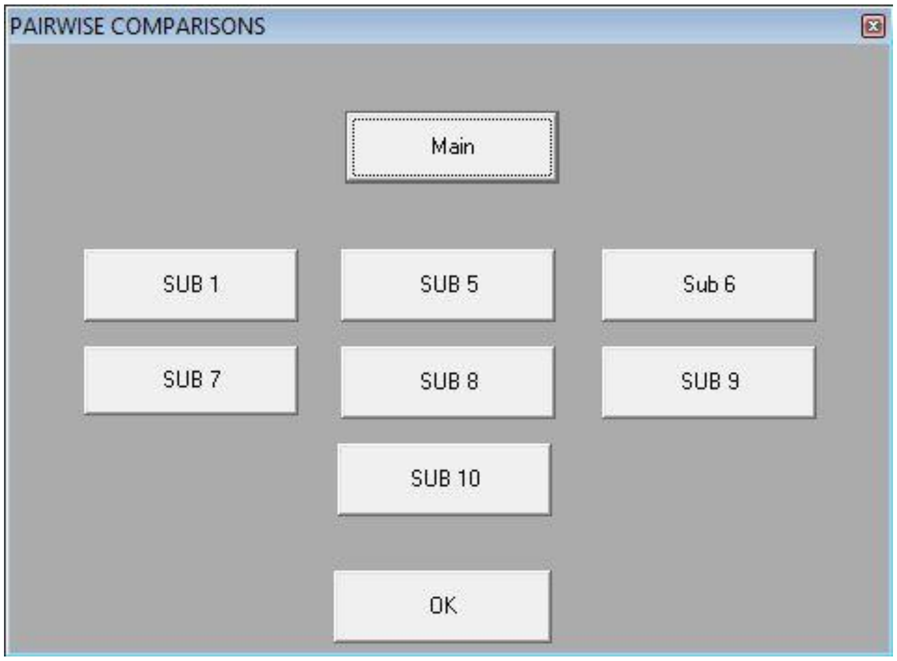
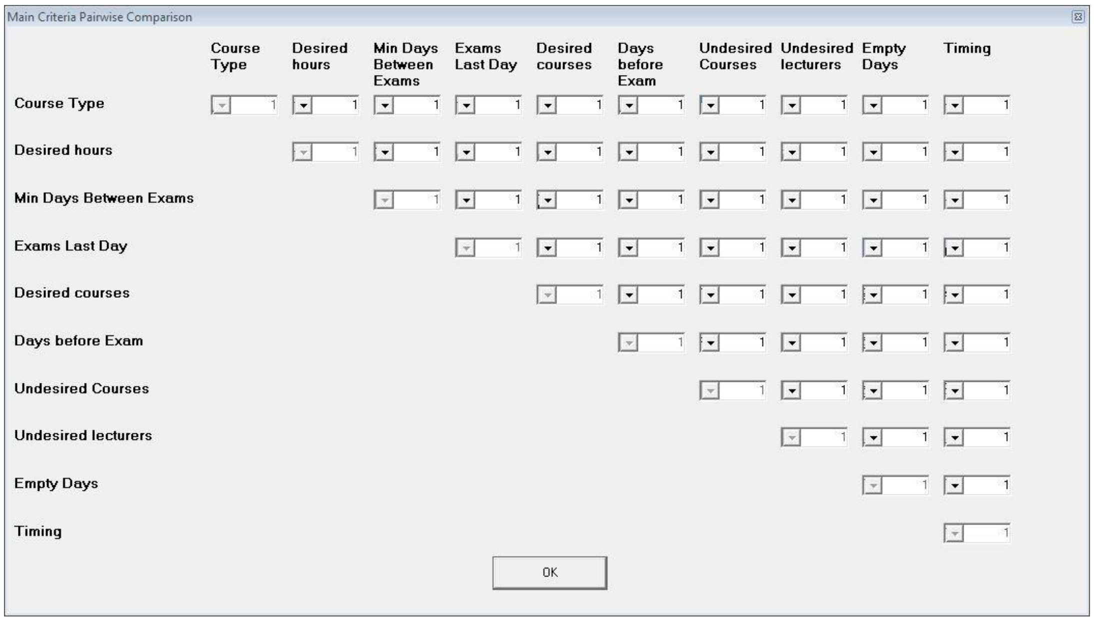


Figure 4.18: AHP Clusters Navigation Form

Figure 4.19 shows the pairwise comparison form of the main criteria.

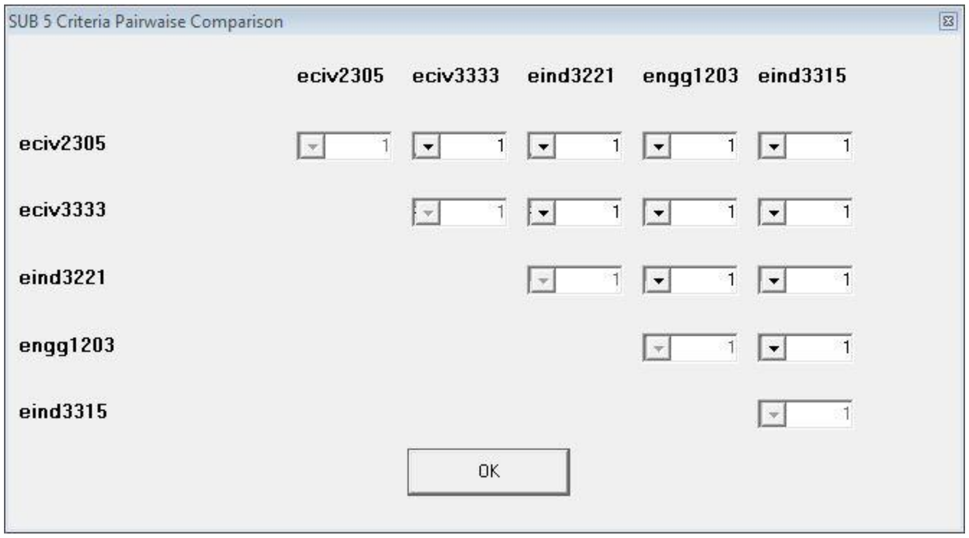


Main Criteria Pairwise Comparison

	Course Type	Desired hours	Min Days Between Exams	Exams Last Day	Desired courses	Days before Exam	Undesired Courses	Undesired lecturers	Empty Days	Timing
Course Type	1	1	1	1	1	1	1	1	1	1
Desired hours		1	1	1	1	1	1	1	1	1
Min Days Between Exams			1	1	1	1	1	1	1	1
Exams Last Day				1	1	1	1	1	1	1
Desired courses					1	1	1	1	1	1
Days before Exam						1	1	1	1	1
Undesired Courses							1	1	1	1
Undesired lecturers								1	1	1
Empty Days									1	1
Timing										1

Figure 4.19: Main criteria Pairwise comparison matrix

Figure 4.20 shows the pairwise comparison form of the desired courses sub-criteria.


SUB 5 Criteria Pairwaise Comparison

	eciv2305	eciv3333	eind3221	engg1203	eind3315
eciv2305	1	1	1	1	1
eciv3333		1	1	1	1
eind3221			1	1	1
engg1203				1	1
eind3315					1

Figure 4.20: Desired courses sub-criteria Pairwise comparison matrix

Either in the main pairwise comparison form or the other sub-criteria forms, Comparison will be conducted only between elements that were set by the user in the criteria section in the application interface. Also, weights resulted from AHP will be distributed among them. The software is programmed such that once the AHP button is clicked, combo boxes matrices will be generated in all of the pairwise comparison forms, each one has a dimension equal to the number of elements set by the user in it is associated cluster. Also, both dimensions will be named after these elements. The combo box that falls at the intersection of a certain element row and another's column represents the relative value of the first with respect to the last. These combo boxes are filled upon their creation with values from 9 to 1/9, however the default value will be one. That is why in case a matrix was not manipulated, then all of its elements will get the same weight of importance with respect to each other.

4.3.4 Model Document

The model document is a text file created at the same location of the application once it is launched. The function of this file is to contain the model while it is constructed gradually. Once the model is complete, this file will be converted into Lp format file that is recognized by the solving package used to solve the model.

This file is also good in case other solvers rather the one used in this system is used to solve the problem in order to generate other solutions or to fasten the solving process.

The model document can be opened any time through notepad or any other text application either manually by going to its path on the computer, or simply by clicking the button named “Open File” in the control section of the software (Figure 4.21).

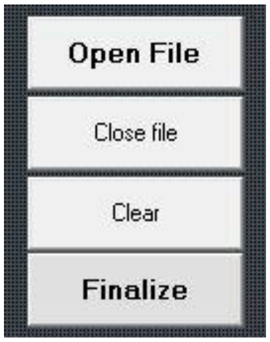
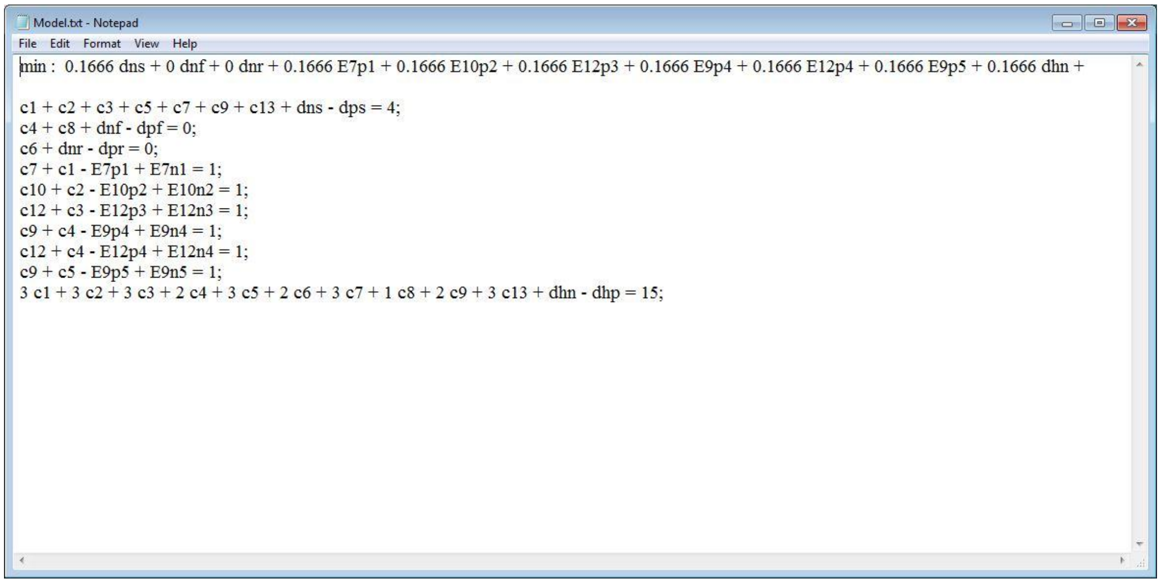


Figure 4.21: Model Document Controls

The model document is first created with the word “min:” in it as a start, since the objective function will always be a minimization, it also takes into consider the required format of LP files. A function called Write–To–File is developed in the software which continuously appends to this file. This function is a bit complicated because writing to text files in visual basic necessitate reading and rewriting of the entire file each time a modification is needed since supplements is continuously needed at the end of the file represented in new constraints and then another is also needed back in the objective function in the first line represented in deviation variables. Moreover, it is necessary to keep the format of LP files. Each time a construct button is pressed in any of the various criteria in the criteria section, all associated constraints to that criterion that were discussed earlier in the theoretical section and the appropriate deviation variables will be appended to the model using this function. A preview of the model document as it is in progress is shown in figure 4.22.



```
min :  0.1666 dns + 0 dnf + 0 dnr + 0.1666 E7p1 + 0.1666 E10p2 + 0.1666 E12p3 + 0.1666 E9p4 + 0.1666 E12p4 + 0.1666 E9p5 + 0.1666 dhn +

c1 + c2 + c3 + c5 + c7 + c9 + c13 + dns - dps = 4;
c4 + c8 + dnf - dpf = 0;
c6 + dnr - dpr = 0;
c7 + c1 - E7p1 + E7n1 = 1;
c10 + c2 - E10p2 + E10n2 = 1;
c12 + c3 - E12p3 + E12n3 = 1;
c9 + c4 - E9p4 + E9n4 = 1;
c12 + c4 - E12p4 + E12n4 = 1;
c9 + c5 - E9p5 + E9n5 = 1;
3 c1 + 3 c2 + 3 c3 + 2 c4 + 3 c5 + 2 c6 + 3 c7 + 1 c8 + 2 c9 + 3 c13 + dhn - dhp = 15;
```

Figure 4.22: Model Document in Progress

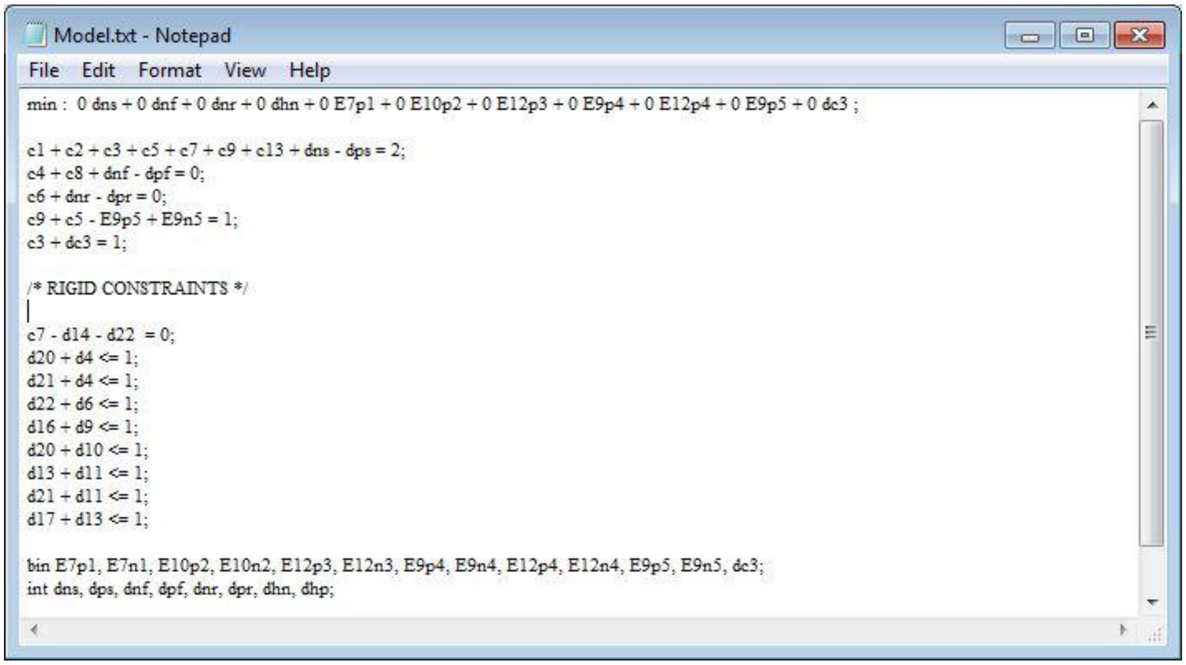
The function of the button named “Close File” shown in figure 4.18 is to close the model document in case it was opened. Both “Open File” and “Close File” buttons are provided only for demonstration purposes. They are also necessary for the operation research analyst to keep monitoring the development of the model.

The button named “Clear” is responsible of wiping the entire model document and prepare it again to be filled. It should be pressed in case the user needs to construct a new problem with different input with respect either to the criteria involved or the weights of importance.
The last button in the model document controls is called “Finalize” (Figure 4.18). it was called so because it is responsible of setting the model document in its final shape that is ready to be solved.

Finalize has four functions as follows:

1. Add the hard constraints.
2. Define all variables involved in the model. Each variable will be declared with its appropriate type.
3. Set the model in its final shape which format is of LP file.
4. Make a copy of the model document with “LP” extension, so to be solved using the solving package.

Figure 4.23 shows the model document upon finalizing.



```
min :  0 dns + 0 dnf + 0 dnr + 0 dhn + 0 E7p1 + 0 E10p2 + 0 E12p3 + 0 E9p4 + 0 E12p4 + 0 E9p5 + 0 dc3 ;

c1 + c2 + c3 + c5 + c7 + c9 + c13 + dns - dps = 2;
c4 + c8 + dnf - dpf = 0;
c6 + dnr - dpr = 0;
c9 + c5 - E9p5 + E9n5 = 1;
c3 + dc3 = 1;

/* RIGID CONSTRAINTS */

c7 - d14 - d22  = 0;
d20 + d4 <= 1;
d21 + d4 <= 1;
d22 + d6 <= 1;
d16 + d9 <= 1;
d20 + d10 <= 1;
d13 + d11 <= 1;
d21 + d11 <= 1;
d17 + d13 <= 1;

bin E7p1, E7n1, E10p2, E10n2, E12p3, E12n3, E9p4, E9n4, E12p4, E12n4, E9p5, E9n5, dc3;
int dns, dps, dnf, dpf, dnr, dpr, dhn, dhp;
```

Figure 4.23: Finalized Model Document

4.3.5 Solving

The solver used in DSSPS is a freeware package named Lp-solve. Lp_solve is a linear (integer) programming solver based on the revised simplex method and the Branch-and-bound method for the integers. Lp_solve was originally developed by Michel Berkelaar at Eindhoven University of Technology.

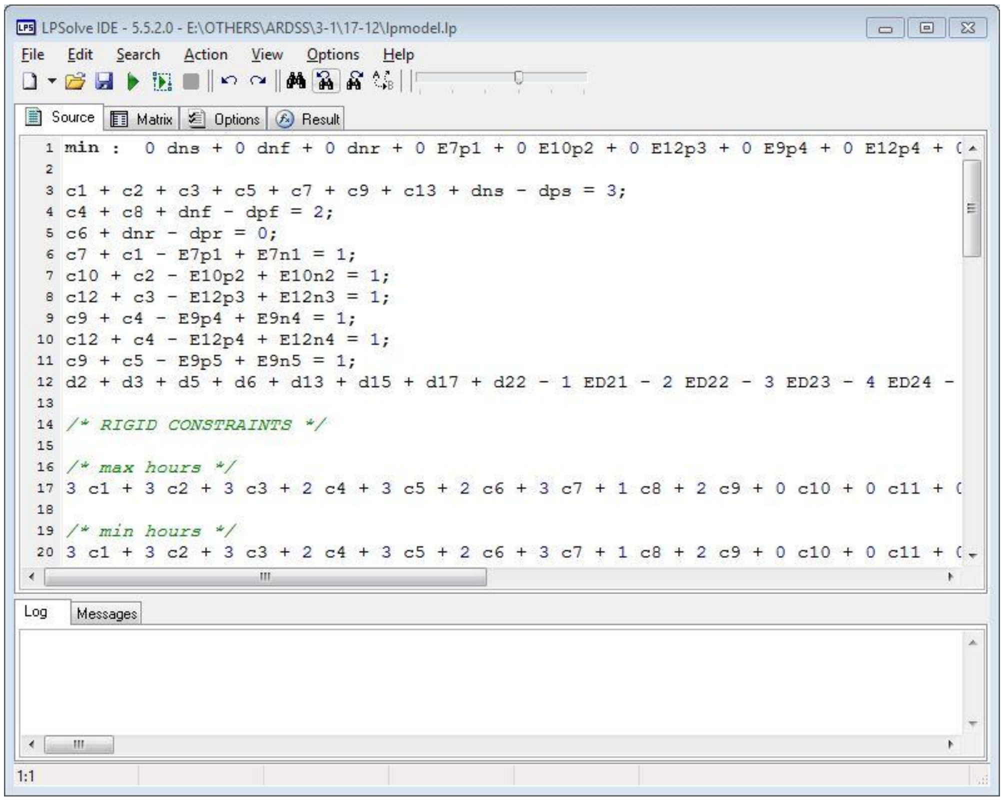
From its name Lp-solve do not have the capability of solving models that contains equations of the second order or higher, however, other commercial solving packages do. Nevertheless, there will be no need for them since the model is designed to contain linear equations only.

Lp-solve can also be called as a library from different languages like C, VB, .NET, Delphi, Excel and Java using APIs. The API is a set of routines that can be called from a programming language to build the model in memory, solve it and return the results.

There is now also an IDE program called LP-Solve IDE (Figure 4.24) that uses the API to provide a Windows application to solve models. With this program there is no need to know anything of API or computer programming languages. If a model is provided to the program it will solve the model and give the results.

The following is a list of Lp-solve features:

- Mixed Integer Linear Programming (MILP) solver
- Basically no limit on model size
- It is free and with sources
- Supports Integer variables, Semi-continuous variables and Special Ordered Sets
- Can read model from MPS, LP or user written format
- Models can be built in-memory without the use of files
- Has a powerful API interface
- Easy callable from other programming languages
- Provides different scaling methods to make the model more numerical stable
- Has pre-solve capabilities to tighten constraints/make the model smaller and Faster to solve
- Has a base crashing routine to determine a starting point
- Allows restart after making changes to the model. Solve continues from the last found solution
- Has the possibility to convert one model format to another format
- Provides post-optimal sensitivity analysis.


LPSolve IDE - 5.5.2.0 - E:\OTHERS\ARDSS\3-1\17-12\lpmodel.lp
File Edit Search Action View Options Help
Source Matrix Options Result
Log Messages
1:1

```
min :  0 dns + 0 dnf + 0 dnr + 0 E7p1 + 0 E10p2 + 0 E12p3 + 0 E9p4 + 0 E12p4 +

c1 + c2 + c3 + c5 + c7 + c9 + c13 + dns - dps = 3;
c4 + c8 + dnf - dpf = 2;
c6 + dnr - dpr = 0;
c7 + c1 - E7p1 + E7n1 = 1;
c10 + c2 - E10p2 + E10n2 = 1;
c12 + c3 - E12p3 + E12n3 = 1;
c9 + c4 - E9p4 + E9n4 = 1;
c12 + c4 - E12p4 + E12n4 = 1;
c9 + c5 - E9p5 + E9n5 = 1;
d2 + d3 + d5 + d6 + d13 + d15 + d17 + d22 - 1 ED21 - 2 ED22 - 3 ED23 - 4 ED24 -

/* RIGID CONSTRAINTS */

/* max hours */
3 c1 + 3 c2 + 3 c3 + 2 c4 + 3 c5 + 2 c6 + 3 c7 + 1 c8 + 2 c9 + 0 c10 + 0 c11 +

/* min hours */
3 c1 + 3 c2 + 3 c3 + 2 c4 + 3 c5 + 2 c6 + 3 c7 + 1 c8 + 2 c9 + 0 c10 + 0 c11 +
```

Figure 4.24: LP–Solve IDE

Although Lp–solve supports models to be built in memory so that to avoid the use of text files. It was necessary to use a text file for monitoring purpose, so that to continually examine the generation code as it was built, besides, this text file will be useful in case LP solving engines other than LP–Solve were used.

Figure 4.25 shows the group of buttons related to the final process in the software represented by solving and introducing the solution.

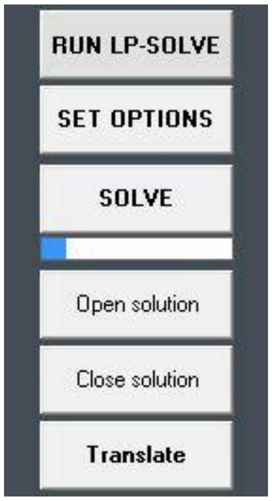


Figure 4.25: LP–Solve IDE

Once the model document is finalized, it is now ready to be solved. The button named "RUN LP–SOLVE" will launch the LP–SOLVE IDE in case it was installed on the computer. It is going to run the file named "Model.lp" which was created by the "Finalize" process. Thus, LP–SOLVE IDE will open with the same model file opened through it and is just ready to be solved (Figure 4.24) either by clicking on the "solve" button up in the toolbar or by pressing "F9".

However, the software does not need the IDE to be installed, since it is going to solve the model using the application programming interface functions of the LP–solve library. This library will be included in the software package so to be installed with the software to any machine. Nevertheless, running the IDE will still be important for the sake of a deep analysis of the solving process. It may help OR/MS experts to review the iterations of the optimization process, get other information about the solving process and the resulting solution or to conduct sensitivity analysis.

To use the LP–SOLVE APIs to just solve the model silently, the user can click the button named "SOLVE".

Prior to solving, this “Solve” button will create an output file called “output.txt” and set it to contain the resulting solution.

Solving time varies according to the complexity of the model. It also depends on the computer processor. A model complexity is determined by the number of variables and constraints that exist. The number of variables and constraints depend mainly on three factors:

A. The number of available courses and sections in the related database and their characteristics.

   The previous factor relates to the number of decision variables which will increase as the size of the available classes’ database increases. Moreover, the characteristics of these classes determine the number of constraints built due to lectures timing conflicts or final exams conflicts in both the soft and the hard section.

B. Criteria involved by the user.

   The type of criteria involved affects the number of constraints built in the soft section, because some certain criteria produce more constraints than others as discussed earlier in the theoretical design section.

C. Goals set by the user.

   This factor applies particularly in both the minimum number of days that separates between final exams and the minimum number of empty days that precedes a certain course final exam. The higher this number is, the more constraints will be built.

The problem is that unlike the IDE platform there will be no sign whether the solving process is yet finished or not. The software will just hang during this period. To solve this issue, a progress bar was created that is periodically refilled. This progress bar will hang during the solving process and rerun again when it is finished.

Once the solving process is over, results will be printed in the output file mentioned earlier. Results include the value of the objective function, besides, the values of all engaged variables. However, only sections decision variables are the ones that matter. Those decisions variables that were assigned the value of one represent the group of classes that should be taken as a final result of the system. Note that the word “Class” is used to denote a course, timing and a lecturer.

Solving Options

Many commercial solving packages support presenting multiple optimum solutions for a certain linear programming problem. Other optimum alternatives represent other solutions that have the same optimum objective function value with different decision variables ones.

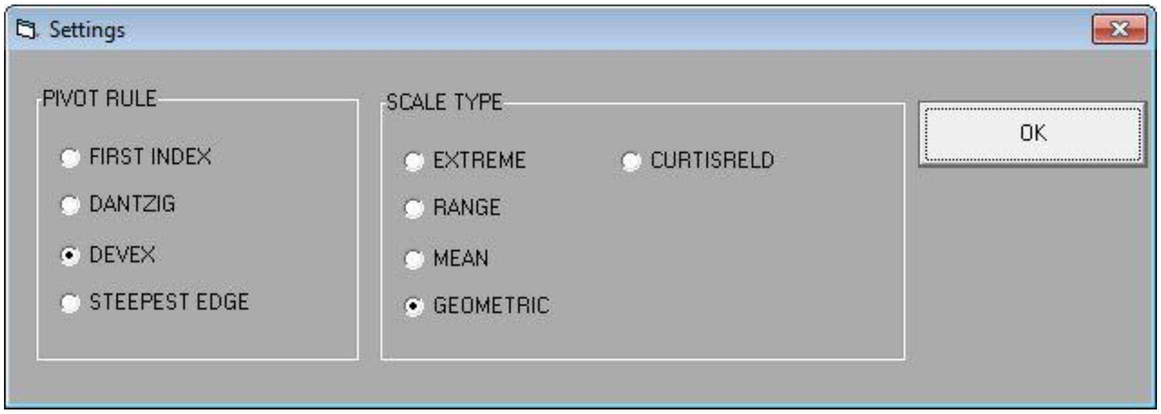
This feature is not supported in LP-Solve. LP-Solve will just present the first optimum solution it finds. It will search no more for alternatives. This is because of the nature of the branch and bound algorithm that LP-Solve follow to solve an integer problem.

However in fact, one should think whether this feature really matters. Since the resulting solution satisfies the user targeted goals to the maximum extent. Why should he search for alternatives?

If the user is not satisfied with the resulting solution, he can simply redefine his goals and priorities then solve the problem again.

Nevertheless, as previously stated, this system allows for problems to be solved using other solving packages that may support this feature since it creates a model file with LP format. This format can be transformed to any other using the LP–Solve IDE. This file can be taken to be solved by the intended solving package, then, result can be returned to be translated using this software as shown next in the solution section.

Moreover, a solution to this matter is presented in this software through solving settings. Solving settings represent the ways by which LP–Solve approaches the solution. By clicking the button named "Set Options" a window shown in figure 4.26 will show up that permits the user to manipulate solving settings. Changing these options may cause – but not necessarily – the optimization process to reach another optimum solution.



PIVOT RULE: FIRST INDEX, DANTZIG, [✓ DEVEX], STEEPEST EDGE

SCALE TYPE: EXTREME, CURTISRELD, RANGE, MEAN, [✓ GEOMETRIC]

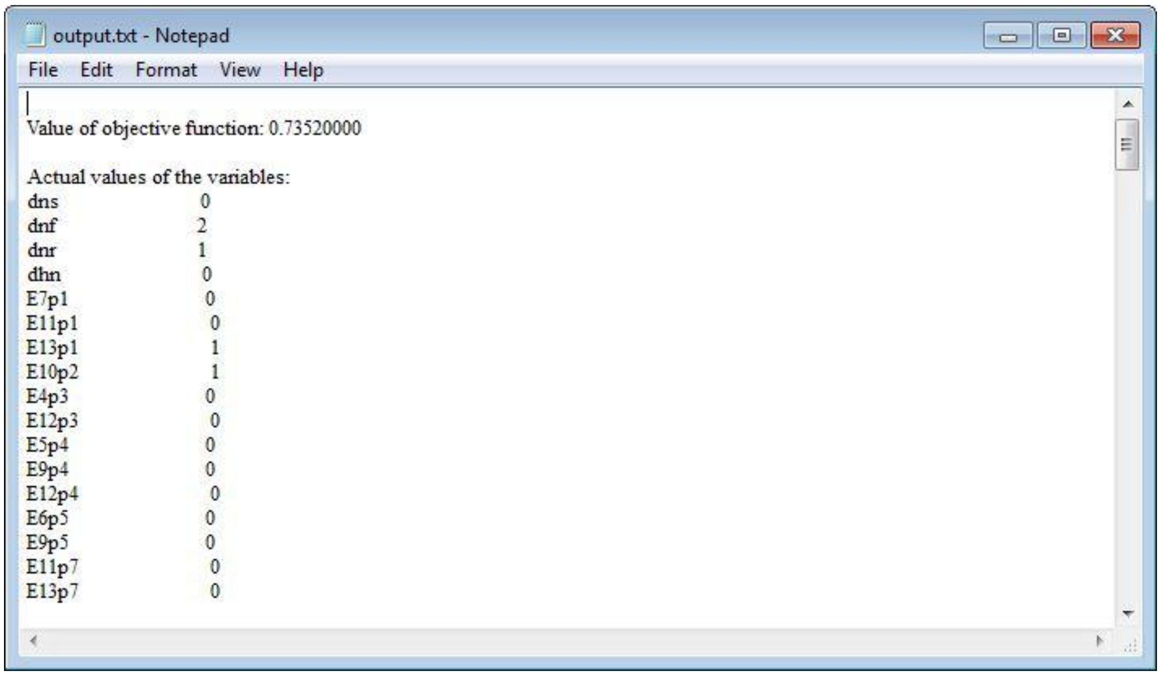
Figure 4.26: LP–Solve settings

4.3.6 The Solution

A main feature of this system is that it is designed to provide a tool by which non-experts or those who have no OR/MS background are allowed to construct their own models and use them to support their decisions. There will be no need for an analyst to act as an intermediary between the system and the decision makers as most of the OBDSSs do. As stated earlier, most of the procedures involved so far in the software are purely for demonstration purposes as this system is still to be introduced. Such procedures may mean something to an operation research specialist but non to a first year student.

The function of the two buttons named “open solution” and “close solution” is to open and close the output text file in which the solution is printed. The values of both the objective function and all involved variables as a final result of the optimization process are included in this file. This file helped the author during the process of designing the system but again may not be of great value to the user. An example of the output file is presented in figure 4.27.

To an OR/MS analyst, value of the objective function represents the summation of deviations from the targeted values of each considered criteria according to weights assigned to each one of them. In other words, the smaller this value the closer the resulting solution from the user targeted set of goals. In case value of the objective function was zero, this means either that all of the user goals are met or that the user did not set any goals.



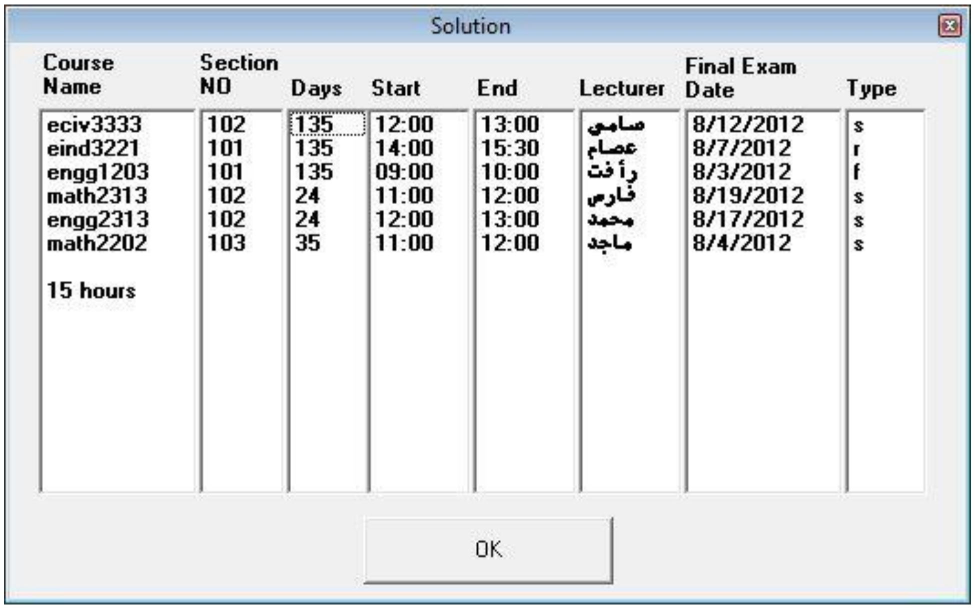
```
Value of objective function: 0.73520000

Actual values of the variables:
dns                    0
dnf                    2
dnr                    1
dhn                    0
E7p1                   0
E11p1                  0
E13p1                  1
E10p2                  1
E4p3                   0
E12p3                  0
E5p4                   0
E9p4                   0
E12p4                  0
E6p5                   0
E9p5                   0
E11p7                  0
E13p7                  0
```

Figure 4.27: LP–Solve IDE

Thus, the final step of the software is to translate the output file in a form that is familiar to the student eyes. Once the button named “translate” is clicked, the output file will be read line by line. Finally, according to the database of the available classes, a list of the sections that should be registered will be viewed in a new window along with all of their related information which is:

A. Name of the course
B. Section number
C. Days in which a sections meet
D. meeting timing of a section
E. Lecturer name
F. Final exam date
G. type of the course

Figure 4.28 shows the window in which final results is viewed as a readable schedule.



Course Name	Section NO	Days	Start	End	Lecturer	Final Exam Date	Type
eciv3333	102	135	12:00	13:00	سامي	8/12/2012	s
eind3221	101	135	14:00	15:30	عصام	8/7/2012	r
engg1203	101	135	09:00	10:00	رأفت	8/3/2012	f
math2313	102	24	11:00	12:00	فارس	8/19/2012	s
engg2313	102	24	12:00	13:00	محمد	8/17/2012	s
math2202	103	35	11:00	12:00	ماجد	8/4/2012	s
15 hours							

Figure 4.28: Solution window

4.3.7 Other Used Tools

A. Progress Bars

Progress bars are distributed among the various elements in the criteria section. These bars denote the weight that each element gets as a result from the AHP process.

There are two types of progress bars. The big ones – vertically positioned – represent the main criteria elements, and the small ones – horizontally positioned – represent the sub-criteria elements.

Once the OK button in the AHP clusters navigation form is clicked, both local and global weights will be calculated. At the beginning, progress bars will be assigned local weights, that is, each element will be assigned a weight with respect to its cluster. However, once the button called “calculate final scores” is clicked, progress bars will be assigned its associated elements global weights, resulted from multiplying the local weight of each element by its parent. The overall weights that sum to one will be only distributed among the final level. Final level may contain both sub-criteria and main criteria which have no sub-criteria, these main criteria will keep its local weights as it is, as a result, its associated progress bars value will remain the same. The unbranched main criteria are the desired number of credit hours, the desired minimum number of days between exams and the desired furthest final exams date. Progress bars of the branched main criteria will be assigned a zero as its local weight is now distributed among its sub-criteria.

B. Check boxes

They are also distributed among the various elements in the criteria section. Check boxes are important to denote that a certain element is considered by the user whether it is a main criterion or a sub-criterion, consequently, it will be considered in the AHP process. Check boxes are automatically checked once their associated elements are manipulated. Also, the check box of a branched main criterion will be automatically checked once any of its sub-criteria is checked and unchecked once all of its sub-criteria are unchecked.

C. “X” Buttons

These “X” buttons is associated with all sub–criteria of main criteria number five, seven and eight. Their function is to cancel a selected item of the associated sub–criteria whether it is a desired course, an undesired course or an undesired lecturer.

As discussed before, canceling an item will refill it in all combo boxes lists of the same main criteria and the ones from the opposite criteria. For example, selecting a desired course will subtract it from all other combo boxes lists as well as the undesired courses ones, however, canceling it, will refill this course in all of them again.

4.3.8 User's Guide

This part clarifies how to use the current version of the software. It shows in details steps that should be executed respectively in order to obtain meaningful results.

Login to the Program

Login is important to inquire the proper information of the current user. As for now there will be no need to enter a user name and a password until the software is actually bound to the university database. As stated before, the database used so far is an imaginary one. This database was used for developing purposes.

Define Goals

This step is accomplished through the criteria section of the software interface the way it is discussed before.

Whenever the user manipulates any element in the criteria section, the associated check box will be checked. Check boxes denote which elements will be considered in the AHP process, they also determine which elements constraints are going to be built. If a certain element is no longer considered, user can simply uncheck its associated check box. This step involves a determination of goals but not yet to press the construct buttons till weights is assigned in the next step.

Set Priorities

Upon software startup global and local weights of all elements will be assigned the value of zero. Unless the user enter the AHP clusters navigation window there will be no meaning of the model since coefficients of all objective function variables will be zeros.

Once the user enter the AHP clusters navigation window, elements of the same cluster will be given equal weights, also, global weights will be calculated based on that, even though the user did not enter any of the pairwise comparison forms, however, the user can enter these forms and set customized weights through pairwise comparisons.

An equal weight to elements of the same cluster may be unrealistic, especially in the main cluster, for one thing, some elements such as main criteria that are not widely ramified will obtain a lot more weight than those how are widely ramified – of course it depends on the number of elements considered in each –. For another, the nature of elements considered differs.

That is why pairwise comparisons should be conducted at least for the main cluster, it is also why unit penalized of each criteria – discussed earlier in the theoretical section – should be recalled during this process.

If the user was perfectly consistent, there will be no need to fill the entire matrix; relations of one element with respect to the others will be enough to calculate the weight vector, while the rest of the matrix will be redundant, since:

$$t_{ij} \times t_{jk} = t_{ik} \qquad (\text{for all } i;\ j;\ k = 1,2 \ldots n) \qquad (4.22)$$

However, the user will not be perfectly consistent. On the other hand being perfectly consistent is not necessarily error free. So this is the idea of AHP, which is based on strengthening accuracy through redundancy.
Once the user finishes the pairwise comparisons process and exit the AHP clusters navigation form, global weights will be recalculate and distributed among the considered elements.

Construct the Model

The fourth step is to construct the model by clicking the construct buttons in all criteria sections that contain any considered elements.
It is not important to wait for the progress bars to be filled as weights are already calculated and stored. Also, the order at which these buttons are pressed does not matter. The objective function is just a summation of deviations, besides; the order by which constraints are arranged does not matter in linear programming.

Finalize the Model

By pressing the “Finalize” button which function is discussed earlier.

Solve

In case “Run LP–Solve” is pressed, the IDE platform of LP–Solve will run with the model already opened in it. From there, the user is able to solve the problem, however, will not be able to translate the results unless he understood the theoretical design of the system.

Nevertheless, the “Solve” button that uses the LP–Solve APIs is the one designed for normal users, once clicked, the solving process will begin causing the whole software to hang as well as the running progress bar beneath the solve button. When this bar runs again, it means that solving process has ended and the solution is now already printed in the output file.

Translate the Results

By clicking the translate button. The output file will be translated into a readable schedule form.

4.4 Information Feeding Mechanism

Information feeding mechanism refers to the way by which the software needed information is going to be enquired from the university registration database, reformed in the appropriate shape and delivered to the software as it asks for it.

The software backend database is supposed to be formed to suit the logged in student. It should be emptied from all previous data and refilled with the new user's automatically. This is the vision of the system, a vision that cannot come to reality unless the system is granted permission to access the university registration database and use the appropriate queries. However, till this point, all I am offering is an idea of a DSS, not a ready software package.

4.4.1 Available Classes Query

As stated before, for a course to be considered "Available", it should meet the following conditions:

e. Offered for the current semester.
f. Unstudied before.
g. Their prerequisites are fulfilled.
h. At least, one section of it still available. Any course of which all sections are full should not be listed in the courses table, even though it satisfies all three previous conditions.

On the other hand, every single course in the available courses table should have an associated group of sections listed in the available sections table. These sections should meet only two conditions:

d. Relates to a course in the available courses list.

e. Offered for the current semester.

f. Not full yet.

It should be noted that the software will work with whatever information provided to it, on the other hand, it is illogical to expect realistic results when the input is not, for example, if the backend database included unavailable courses or sections, the output may also include them, furthermore, should the backend database includes wrong data about some sections or courses, it will cause the entire optimization process to be based on wrong data, what will result in either a non-realistic solution or a solution that is not really optimized with respect to the actual data. That is why it is essential for any university that wishes to adopt the system to design the proper queries that guarantees all of the previous conditions. These queries will cause the previously listed five issues to be out of the way as they have been already taken care of.

The process of building this query depends on the schema and diagram of a university registration database. So, it is better to show an example of it instead of a strict approach. An example is the approach followed to reach this information feeding mechanism in the Islamic university of Gaza.

Once the software is ready, its backend database can be replaced with a direct access to the university registration database, however, during the experiment of testing the system in the Islamic university of Gaza, not only they refused to grant the system an authorization to the registration database, but also refused to provide me with the exact query built to retrieve the required data.

Appendix C shows the query used by the Islamic university registration database specialist to fetch the available courses and section for a certain student in a certain semester along with its characteristics required by the software.

It's important to note that "Available courses" in universities that adopt the credit hour system not only refer to courses required by the major but also to any course that a student is allowed to be enrolled in during a certain semester. Even though it is not a part of his/her major. Besides, sometimes a student may decide to repeat a certain course just to have the opportunity of achieving a higher score. This may lead us to categorize the available courses as follows:

A. Basic Available Courses:

   Refer to courses required by the current major. (Available), should be automatically included.

B. Other Available Courses:

   Refer to courses from other majors that a student is allowed to be enrolled in. only included as requested by the user.

C. Repeated Courses:

   Refer to courses that a student wishes to study again.

The system is designed to just include the first type of courses automatically as they are obligatory. The other two types need the student requesting. Nevertheless, these two types may be included later. However, other queries should be developed to prepare tables for both.

Another important point –regarding the first criteria– is that there must be a mechanism by which courses can be classified as university, faculty and department requirements as they are inquired with respect to the student major, because a course may be considered as a university requirement for a student but a department requirement to another.

4.4.2 Data Format.

Data contained in both tables should have a certain form with which the software was designed and is familiar. For example:

A. Time is expressed in the software code as numbers that begins with zero representing 8 o'clock morning, and ends with 20 representing 6 o'clock afternoon, so that each step represents half an hour.

B. Days from Saturday to Thursday are represented by numbers from 1 to 6.

Data retrieved by the Islamic university specialist were not as they are supposed to be –as for the software–. For example, days were not denoted in the Islamic university registration databases the same way as in the DSSPS. Table 4.2 shows the difference between the two.

Table 4.2: Islamic University Days Coding Versus DSSPS

Day	Islamic University System	DSSPS
Saturday	S	1
Sunday	N	2
Monday	M	3
Tuesday	T	4
Wednesday	W	5
Thursday	–	6

Whereas time was represented in 24 hours system with no separation between minutes and hours, the difference between the Islamic university system and the DSSPS is shown in table 4.3.

Table 4.3: Islamic University Timing System versus DSSPS

Time	Islamic University system	DSSPS
08:00 AM	800	0
08:30 AM	830	1
01:00 PM	1300	10
01:30 PM	1330	11

Thus, it was necessary to convert these data to the familiar format recognized by the software. Appendix D shows the functions written in VB to accomplish this task.

4.5 DSSPS Flexibility

A. The number of desired courses, undesired courses, desired lecturers, undesired lecturers allowed in the software.

   Although, it is unlikely for any of the previously mentioned criteria to exceed eight, this limit can be extended easily through a few modifications of the software code and interface.

B. Lectures may be held at days from Saturday to Thursday.

   At any university, there must be at least one day off, however, the only problem that remains is which one it is. This problem can be simply solved by only altering names of the days referring slots in the ninth and the tenth criteria in the software interface according to that day. There will be no need to modify the code because days are represented in it as numbers from one to six rather than letters that represent the first letter of each day. Another thing that will also need to be modified is the way by which characters denoting days retrieved from the university database will be transformed to the software recognizable format.

In case a university had more than one day off, then, the missed day in the software should be assigned to one of them, there will be no problem with the other empty days, the generation process will simply realize that there are no lectures in it, as a result, no constraints will be built regarding the ninth and the tenth criteria for these days.

C. This system has the ability to be used in any university that uses the credit hours system, however, maybe with slight modifications.

D. Even if this software was not imbedded within a university site, it still can be used by students privately as a desktop application. The university will just has to deliver the required data regarding the available sections once asked by a student in the form of an access database file. Then, the student can only place it in the right path on his PC and run the software.

E. Due to the way by which the regular model is formulated and the mechanism by which models are generated, it is too easy to add other criteria that may show up later. All what is needed is to add an additional field to the courses or the sections table of the available classes' database depending on to which table this new criterion or characteristic belongs.

   Also, an additional section must be developed at the software interface that is responsible for generating the appropriate constraints related to the new criterion.

F. What applies to undergraduate students also applies to graduate students who follow the credit hours system since they share the same characteristics and procedures of registration, moreover, the system was designed to cover time from 8 o'clock morning to 6 o'clock afternoon, thus, graduate students lectures which are usually held at late hours are also covered.

4.6 Cost of Application

Such system will not cost the university a lot. Cost can be summarized in the software development costs which should take into account a fancy, easy and effective interface besides taking care of the security issues using a modern and a strong programming language. The second thing is a set of servers with super specifications that qualify it to serve multiple users spontaneously

4.7 DSSPS Assumptions

4.7.1 Assumptions Used To Develop the Software

A. Classes held at multiple days are at the same timing.

B. Course information of a certain course (such as the number of credit hours, course type, final exam timing) is the same for all of its sections.

C. For a student, desired courses, undesired courses, desired lecturers or undesired lecturers will not exceed eight per semester.

D. Lectures may be held at days from Saturday to Thursday.

E. A student may set a minimum period to precede a certain course final exam only if this course is desired.

F. It's uncommon for a student to have more than one undesired period within the same day.

4.7.2 General Assumptions of the System

A. Offered courses, offered sections, sections timing, final exams timing or lecturer's allocation are supposed to be prepared by the university as they will be treated by the software as fixed information.

B. University site should provide the software with a database that contains the required information with the required form.

C. The student is well aware about what courses is best for him at the current time to avoid problems in the upcoming semesters. (This may be also the rule of the academic advisor).

Chapter 5: RESULTS AND ANALYSIS

This chapter shows the process of validating the software through testing. Testing phase involves the process of experimenting the software in an environment in which the software is designed to work.

Software testing can be stated as the process of validating and verifying that a computer program, application or product:

- Meets the requirements that guided its design and development.
- Works as expected.
- Can be implemented with the same characteristics.
- Satisfies the needs of stakeholders.

Depending on the testing method employed, software testing can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed.

The developing stage of the software took about four months. During that period, the software was not developed, tested or seen by anyone, –testing mentioned here means the ongoing testing process of the code as it was developed using a virtual database–.

This is the main reason why testing phase meant a lot. This phase was not only necessary to investigate stakeholders' satisfaction but more importantly to ensure its functionality within the outside environment.

As mentioned before, the experiment of testing the system in the Islamic university of Gaza faced a few obstacles. Well, as a start, the software was not granted an authorization to the registration database. It meant that the system will not be able to directly fetch its required data nor ask for filling its backend database.

The only Approach that appeared suitable to overstep this problem at that time is to fetch data for a group of students picked randomly from different departments and levels so that these students will experiment the software one by one. Each time the backend database will be filled with the proper student data. Thus, testing approach steps could be arranged as follow.

5.1Testing Approach

The first step was to have the registration database specialist from the Islamic university design the available classes query shown in chapter four. This query is responsible for fetching the exact data required by the software for a certain student in a certain semester.

The second step would be to start fetching data for random students using the designed query. The information retrieved by that query represents a data table that includes all available sections of all available courses for a student. This information are supposed to include all what is needed for the software, nevertheless some unimportant information was also included.

The third step is to retain this information which is exported as excel files by copying them to the testing computer –Testing computer is the computer on which testing process will be conducted using the software–, after that, these files will be classified, transformed into Access databases and finally adjusted to the proper format recognized by the software (as discussed earlier in chapter four).

The fourth step is to start experimenting the software with each student one by one. Each time the backend database will be replaced with the proper one that matches the experimenting student.

Again, this approach appeared to be very exhausting because of the following:

A. It is not easy to randomly pick students in order to participate in such a long and complicated process.
B. It takes the database specialist too much time to fetch data for a certain student, moreover, this specialist will not be always available nor have the time.
C. The process of copying, transforming, adjusting data files also take too much time.

All previous points made it almost impossible to follow this procedure. So, instead of gathering specific databases for specific students then, conduct private testing processes, it would be better to retrieve a group of databases that represent a group of departments and/or levels. Once transformed and adjusted, these databases will be suitable to be applied to whatever student who matches that database related department and level. Therefore, one database may fit a large number of students.

However, it may be important to address another point here. It is meaningless to conduct a test regarding a certain student in a certain department, then retest the software for the same student after he changed his department because the student's style choosing his criteria, goals and priorities will not change upon changing the department, on the other hand, the way by which the software work will not differ upon changing the available classes list.

For that, it makes more sense to increase the number of the testing students rather than trying to achieve a specialization diversification.

So, it may be possible to conduct this testing process using a group of a real or imaginary databases even though it does not match the users specializations. In another word, these users – who may also be graduates, not just students – will be asked to imagine that database as if it represents what is available for them currently, they will assume a group of goals regarding this database, assign hypothetical priorities, then, let the software find the best schedule for that data. On the other hand, this user will construct a schedule manually, taking into account the goals and priorities he has already defined while taking caution not to violate the registration regulations.

5.2 Testing Goal

Obviously, the main goal of the testing process is to compare the schedule prepared by the student manually with that one optimized by the software. This goal necessitates existence of criteria to be used for comparison.

Customer satisfaction could be a reasonable criteria, however, it cannot be presented as a strong scientific evidence that corroborate the software, besides, customer satisfaction cannot be quantified, as a result, the advantage of this software over the manual scheduling will not be measurable.

Moreover, the user being unsatisfied with the resulting optimized schedule will usually be a result of an inaccurate determination of his priorities. For example, usually the user will unintentionally place too much weight to a certain goal with respect to the other group of goals that interest him; as a result, he will get a schedule that only satisfies that goal while ignoring all others. This happens because the optimization process only cares about minimizing the objective function value which represents the final penalization score. This score is mainly driven by the penalization factor – or weight – of each goal.

So, testing process cannot rely on user satisfaction, the software has no idea what the user exactly wants, it will only deal with whatever input entered by him. Till the user master usage of the software, testing process will assume that whatever input entered by him describes exactly what he wants.

Thus, a criterion that could be used in such case as a fair and a quantifiable index is the objective function value which should be calculated for both the optimized and the manually prepared schedule. A user inaccuracy defining his priorities will not be a matter anymore, since the resulting inaccurate weights will be applied equally in both the original model and the model used to evaluate the manual schedule.

5.3 Software Testing Part

Therefore, another part was added to the software that permits the tester to form a schedule manually from the current available classes' database.

Figure 5.1 shows a group of buttons used in the software interface that belong to the manual scheduling part.

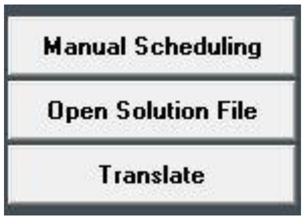
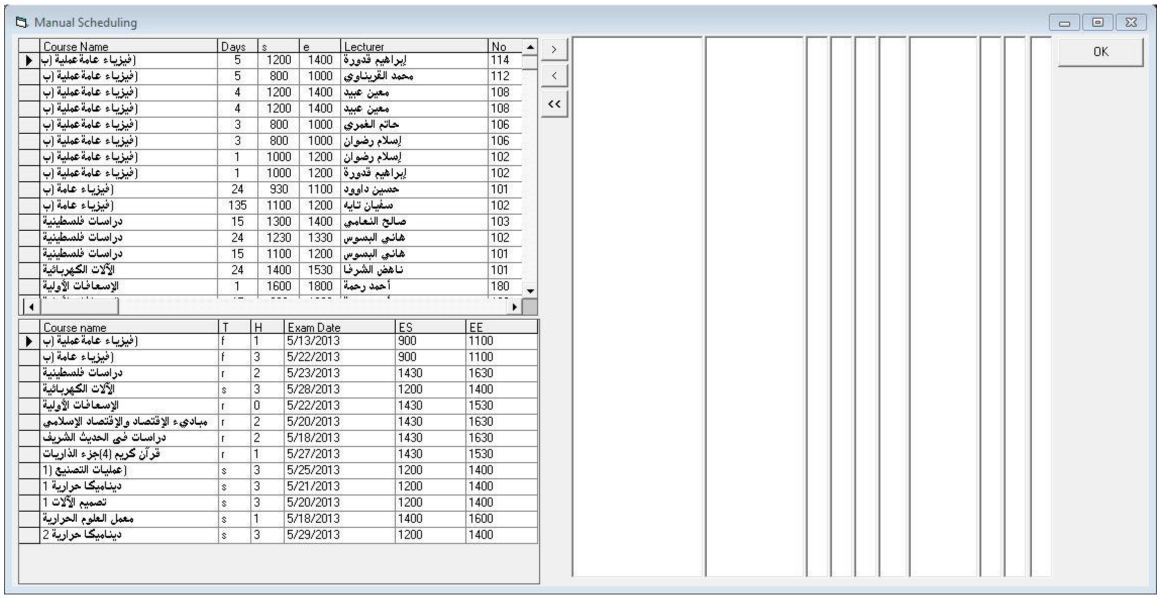


Figure 5.1: manual scheduling buttons

The first button leads to the manual scheduling part shown in figure 5.2. Through this window, the user will be able to form his schedule the same way he does on his page at the university website. The manual scheduling window is divided into two parts, two data tables that represent the two courses and sections tables of the database are located in the first part the same way as in the interface. The other part includes an empty list that contains the chosen classes. In between, a group of buttons that add and remove classes to that list.

An important feature that worth mentioning is that this window is programmed to prevent lectures and final exams timing conflicts, once a certain class is added, there will be no way to add another class that has a timing conflict with the that one either for the lecturers or the final exams. This feature will continuously check for conflicts as the classes are being added, it will also inform the user as with which class this conflict happened, in addition to the cause of this conflict whether it is due to lectures or final exams timing. This will certainly be helpful to avoid such conflicts during the process of manual scheduling, eventually, it is meaningless to evaluate an infeasible schedule.

The final stage is to evaluate this schedule by trying to calculate its corresponding objective function value.



Course Name	Days	s	e	Lecturer	No
(فيزياء عامة عملية (ب	5	1200	1400	إبراهيم قدورة	114
(فيزياء عامة عملية (ب	5	800	1000	محمد القريناوي	112
(فيزياء عامة عملية (ب	4	1200	1400	معين عبيد	108
(فيزياء عامة عملية (ب	4	1200	1400	معين عبيد	108
(فيزياء عامة عملية (ب	3	800	1000	حاتم الغمري	106
(فيزياء عامة عملية (ب	3	800	1000	إسلام رضوان	106
(فيزياء عامة عملية (ب	1	1000	1200	إسلام رضوان	102
(فيزياء عامة عملية (ب	1	1000	1200	إبراهيم قدورة	102
(فيزياء عامة (ب	24	930	1100	حسين داوود	101
(فيزياء عامة (ب	135	1100	1200	سفيان تايه	102
دراسات فلسطينية	15	1300	1400	صالح النعامي	103
دراسات فلسطينية	24	1230	1330	هاني البسوس	102
دراسات فلسطينية	15	1100	1200	هاني البسوس	101
الآلات الكهربائية	24	1400	1530	ناهض الشرفا	101
الإسعافات الأولية	1	1600	1800	أحمد رحمة	180

Course name	T	H	Exam Date	ES	EE
(فيزياء عامة عملية (ب	f	1	5/13/2013	900	1100
(فيزياء عامة (ب	f	3	5/22/2013	900	1100
دراسات فلسطينية	r	2	5/23/2013	1430	1630
الآلات الكهربائية	s	3	5/28/2013	1200	1400
الإسعافات الأولية	r	0	5/22/2013	1430	1530
مبادىء الإقتصاد والإقتصاد الإسلامي	r	2	5/20/2013	1430	1630
دراسات في الحديث الشريف	r	2	5/18/2013	1430	1630
قرآن كريم (4)جزء الذاريات	r	1	5/27/2013	1430	1530
(عمليات التصنيع (1	s	3	5/25/2013	1200	1400
ديناميكا حرارية 1	s	3	5/21/2013	1200	1400
تصميم الآلات 1	s	3	5/20/2013	1200	1400
معمل العلوم الحرارية	s	1	5/18/2013	1400	1600
ديناميكا حرارية 2	s	3	5/29/2013	1200	1400

Figure 5.2: manual scheduling window

5.3.1 Manual Schedule Evaluation

Thus, the main problem here is how to calculate the corresponding objective function value of this manually prepared schedule. In another words, how to formulate the right model of such case?

The first thing we should keep in mind to solve this issue is that the objective function of this model is supposed to be subject to the goals that are already defined by the user, it means that if the soft constraints representing the various goals are violated due to the classes already selected by the user then the value of the objective function should increase with respect to the magnitude of that violation and its corresponding penalization weight, thus, the same objective function, weights and constraints used in the original model should also be used in the manual schedule evaluation model.

So, what is the difference between the original model and the model used to evaluate the manual schedule?

The user being choosing a group of classes is like forcing the optimization process to contain these classes in the final suggested schedule. In another word, the corresponding decision variables of these classes should be assigned the value of one prior to the beginning of the optimization process.

This can be simply accomplished by adding a group of constraints to the model. These constraints are usually equalities that assign the value of one to all of the manually chosen classes in order to ensure selection of these classes (equation 5.1).

$$d_i = 1 \tag{5.1}$$

Where:

d_i: denotes the decision variable of a selected class.

But what about the other classes?. The process of evaluating the manually prepared schedule can be describe as a normal multi-equation with multi-unknown problem Solving process rather than an optimization process. Since most or all of the decision variables are already set to a value, all is needed is the value of the corresponding penalized deviation variables. However, in some cases the model will perform a slight optimization activity that does not affect the chosen classes. For example, the user may define eighteen as a preferred number of credit hours, however, upon manual scheduling, he selected classes that only sums for 14 credit hours. In that case the optimization process will try to select a group of classes in addition to the ones already selected manually by the user in order to approach the credit hours goal as possible. So, for the sake of a just comparison, this optimization activity should be prevented, the only way to do that is to force avoidance of all other classes that were not selected by the user. Thus, the corresponding decision variables of these classes should be assigned the value of zero prior to the beginning of the optimization process.

This can be simply accomplished by adding another group of equality constraints to the model. These constraints assign the value of zero to all other classes that were not selected by the user in order to ensure neglecting of these classes (equation 5.2).

$$d_i = 0 \tag{5.2}$$

Where:

d_i: denotes the decision variable of a non-selected class.

Since the manual scheduling window already prevents timing conflicts, should the hard constraints be also added?

The answer is yes, because there are other constraints related to regulations that are not included in the manual scheduling window such as the maximum and minimum allowed number of hours besides the constraints responsible for binding classes to their subordinates.

Solving process of the manual schedule model will not take time, the optimization process will not check for alternatives the way as the normal model will, because all of the decision variables are already known.

5.3.2 Software Testing Part: Way of Working

As mentioned before, the manual schedule evaluation model is the same as the original optimization one as for the objective function, its weights, the soft constraints and the hard constraints except for the additional equality constraints that force the selection of the manually selected classes only, so, logically, the original model should be built prior to coming to the manual schedule evaluation phase. Or in another word, steps needed to build the original optimization model should be conducted first. It means that the user should first identify his goals and priorities prior to the manual schedule evaluation phase since they represent the base upon which both the objective function and the soft constraints will be generated.

Once both goals and priorities are identified, the user can press the “construct all” button which will construct the objective function and the soft constraints within the model file and the same within anther file that is meant to contain the manual schedule evaluation model.

At this point, the user will be eligible to enter the manual scheduling window and start selecting classes. Once the user is done forming his schedule he can press the “OK” button. This button will construct both the equality constraints that force inclusion of the manually selected classes, avoidance of all other classes and the hard constraints that guarantee full subordination to the university registration regulations.

The “Translate” button in figure 5.1 will show the manual schedule and the associated objective function value will be displayed on top.

5.4 The Questionnaire

Another obstacle that faced the testing process is that users need a lot of time to understand the idea of the DSS besides goal and method of the testing process. So, it seemed logical to illustrate all of this to a large number rather than doing it one by one, however, there was a difficulty providing that number of users with computers at that very moment so that they would use the software. And even if somehow it was possible to provide that number of computers, users will face a difficulty using the software and again it will be necessary to instruct them one by one, each one will take no less than 20 minutes, thus, the remaining users will not wait.

And so, the only solution that appeared convenient in order to solve this problem is to design some sort of a questionnaire that replaces this long and weary process.

The objective of this questionnaire is to gather all required testing data about a certain user which enables me to conduct the testing process later by myself.

These data is classified into three categories, and so, the questionnaire. These three categories are as follow:

A. Goals.
B. Priorities.
C. A manually prepared schedule.

At the end of the questionnaire, there is a table that contains the suggested available classes for the current semester. The process of filling the questionnaire should be done taking into account that the only available classes are represented by what exist in that table. Defining goals and priorities will logically precedes preparing of the manual schedule, because the manual schedule is supposed to be formed based on what goals and priorities the user has in mind.

The goals section imitates the software interface in which the ten criteria are addressed. The user will only specify goals regarding the set of criteria that interest him.

The prioritizing section contains a relationship diagram for the main ten criteria. The user is supposed to conduct the pairwise comparisons method among the various criteria that interest him. To clarify this method to those who are not familiar with it, a simple example is first demonstrated that illustrates it. If part or the entire diagram was left blank, then – as stated in the questionnaire – the associated criteria will be given the same importance (only if they were considered in the first place).

For the sake of simplicity and shortening the time, this prioritizing section will not go further to the sub-criteria level.

The final step in the questionnaire is the process of constructing a schedule manually using the classes in the available classes table.

For the sake of an impartial comparison between the system proposed schedule and the manually prepared one, the user should make the utmost effort preparing this manual schedule to comply with his predefined goals and priorities, but more importantly, the user should avoid breaching any of the registration restrictions since it meaningless to calculate the objective function value of an infeasible schedule. Anyway, the software will not calculate it unless the hard constraints were excluded from the model.

5.5 Results

A variety of students shared filling of the questionnaire. Students from the Islamic university of Gaza, Palestine University and Al-Aqsa University, moreover, graduates from other universities shared this process.

5.5.1 Value of the Objective Function

An example of the main results of a test is shown next, represented in goals shown in table 5.1 and priorities set by the user (figure 5.3), the manual schedule prepared (figure 5.4) and the optimized schedule calculated by the software (figure 5.5).

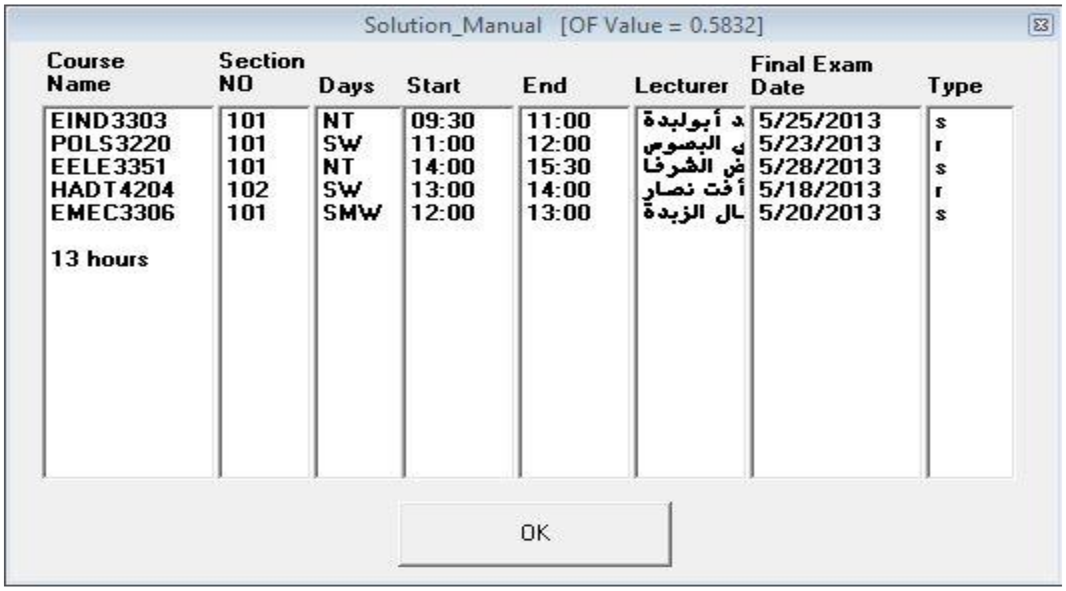
Table 5.1: goals set in test number one.

1	**The desired number of each courses type**	=	3						
		=	2						
		=	1						
2	**The desired range of credit hours**	=	16						
3	**Minimum empty days between final exams.**	2							
4	**furthest date of final exams.**	…							
5	**desired courses. (with or without a desired lecturer)**	EEIE3351	HADT4204	POLS3220					
6	**Number of empty days before a certain course final exam**	…							
7	**The undesired courses.**	EMEC3308	EMEC3111						
8	**The undesired lecturers.**	…							
9	**The desired empty days**	Saturday	Sunday	Monday	Tuesday	Wednesday	THURSDAY		
10	**The desired empty periods**	Saturday	Sunday 10-11	Monday	Tuesday	Wednesday 11-12	THURSDAY		

CRITERIA	1	2	3	4	5	6	7	8	9	10
1. The desired number of each courses type		1	1	1	1	1	1	1	1	1
2. The desired range of credit hours			1	1	1	1	1	1	1	1
3. Minimum empty days between final exams.				1	1	1	1	1	1	1
4. Furthest date of final exams.					1	1	1	1	1	1
5. Desired courses.						1	1	1	1	1
6. days before a certain course final exam							1	1	1	1
7. The undesired courses.								1	1	1
8. The undesired lecturers.									1	1
9. The desired empty days										1
10. The desired empty periods										

Figure 5.3: Pairwise Comparison matrix for the main criteria filled in test number one.

Objective function of the manually prepared schedule of test number one had a value of 0.5832 as shown in figure 5.4.

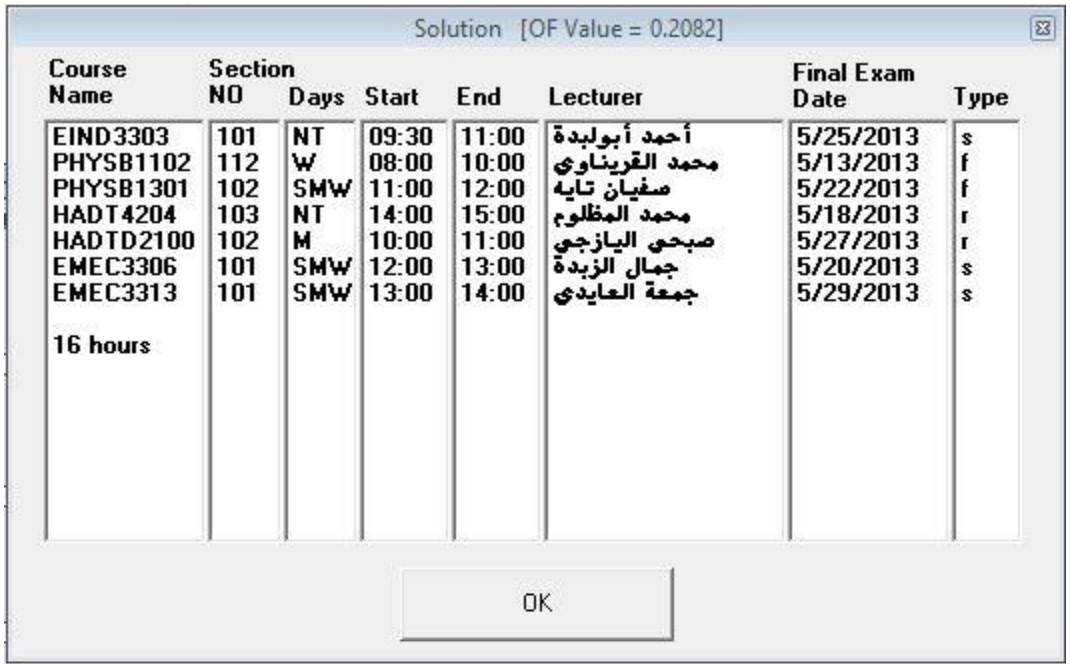


Solution_Manual [OF Value = 0.5832]

Course Name	Section NO	Days	Start	End	Lecturer	Final Exam Date	Type
EIND3303	101	NT	09:30	11:00	د أبولبدة	5/25/2013	s
POLS3220	101	SW	11:00	12:00	ي البصوص	5/23/2013	r
EELE3351	101	NT	14:00	15:30	ض الشرفا	5/28/2013	s
HADT4204	102	SW	13:00	14:00	أفت نصار	5/18/2013	r
EMEC3306	101	SMW	12:00	13:00	ال الزبدة	5/20/2013	s
13 hours							

Figure 5.4: the manual schedule of test number one.

Objective function of the optimized schedule of test number one had a value of 0.2082 as shown in figure 5.5.



Course Name	Section NO	Days	Start	End	Lecturer	Final Exam Date	Type
EIND3303	101	NT	09:30	11:00	أحمد أبولبدة	5/25/2013	s
PHYSB1102	112	W	08:00	10:00	محمد القريناوي	5/13/2013	f
PHYSB1301	102	SMW	11:00	12:00	صفيان تايه	5/22/2013	f
HADT4204	103	NT	14:00	15:00	محمد المظلوم	5/18/2013	r
HADTD2100	102	M	10:00	11:00	صبحي اليازجي	5/27/2013	r
EMEC3306	101	SMW	12:00	13:00	جمال الزبدة	5/20/2013	s
EMEC3313	101	SMW	13:00	14:00	جمعة العايدي	5/29/2013	s
16 hours							

Figure 5.5: The optimized schedule of test number one.

Table 5.2 shows the results of 25 tests conducted in all of the previously mentioned universities. The table includes both objective function values of both the manual and the optimized schedule in addition to the time needed for both.

Table 5.2: Final results of the testing process (objective function value).

Test No	Manual Schedule	Proposed Schedule	Objective Function Value Reduction Percentage
1	0.5832	0.2082	64.30%
2	1.4672	0.6087	58.51%
3	1.6708	0.423	74.68%
4	0.9786	0.7759	20.71%
5	1.8396	0.3022	83.57%
6	1.8666	0.7332	60.72%
7	0.6188	0.3094	50.00%
8	0.7676	0.3906	49.11%
9	0.0738	0.0547	25.88%
10	1.0709	0.6509	39.22%
11	0.7999	0.4666	41.67%
12	0.3292	0.1264	61.60%
13	1.1219	0.3665	67.33%
14	0.6291	0.3509	44.22%
15	1.0275	0.6201	39.65%
16	0.7234	0.2283	68.44%
17	1.0914	0.608	44.29%
18	0.3874	0.056	85.54%
19	0.2483	0.0972	60.85%
20	0.7082	0.1768	75.04%
21	0.8832	0.4999	43.40%
22	0.5829	0.2415	58.57%
23	0.9234	0.4879	47.16%
24	0.9741	0.3519	63.87%
25	1.2912	0.6496	49.69%
Average	0.906328	0.391376	55.1%
Standard Deviation			0.162939

Figure 5.6 is a graph that demonstrates the difference between the objective function value calculated for both the manual schedule and the optimized one.

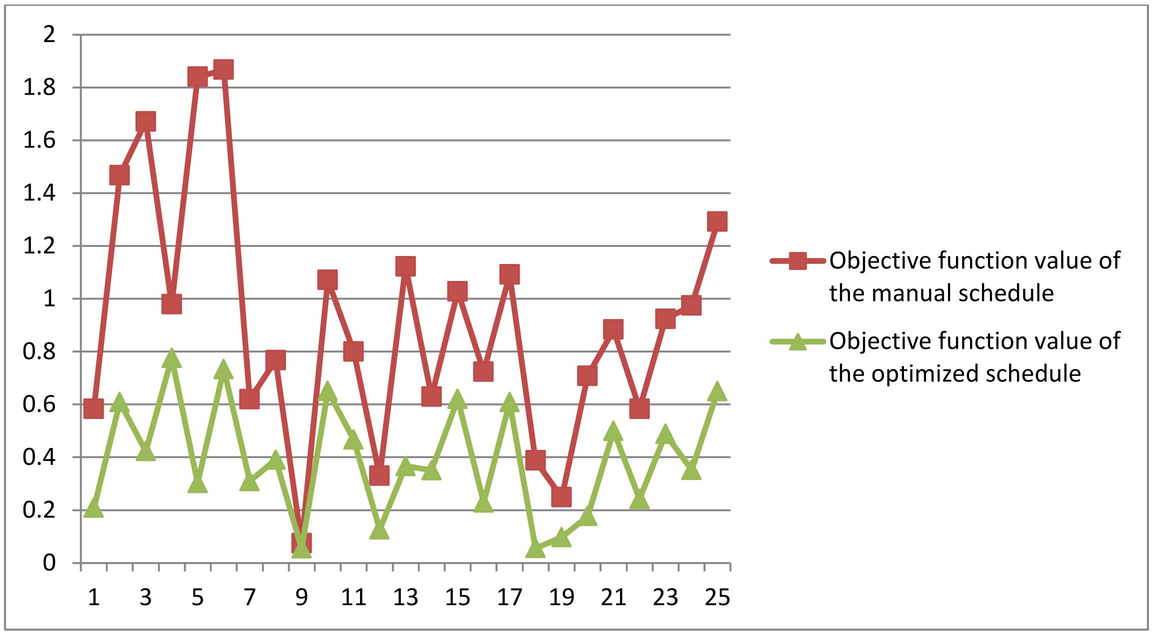


Figure 5.6: OFV's of both the manual and the proposed schedules for all 25 tests.

5.6 Analysis

Naturally, decision making process gets more complicated as both the number of goals and the number of alternatives increase.

Some decisions aim to achieve a single goal while many alternatives exist. Such decisions will be easy to make, because simply one would choose the alternative that best satisfies that single goal.

Similarly, a situation where there is a decision to be made that aims to maximize achievement of a variety of goals while only one alternative exists, such situation cannot be considered a decision making process at all.

In this case, goals are represented by preferences set by the user while alternatives can be measured by the degree to which sections are granted and diversified.

As the number of the available sections and the assigned preferences increase, the multi−objective optimization model generated by the software will be more complicated, worthy to be solved using a computer linear programing solver.

Moreover, when taking about goals that are weighted by their relative importance, it should be noted that having one goal that is assigned a very high weight with respect to the other goals is similar to the case where there is only one goal. In other words, a decision will be too easy to be made –exactly what happened in test number nine.

By easy I mean that a student may not need to use this DSS at all, because his manually prepared schedule will most likely be evaluated as being too close to the optimized one proposed by the system.

Test number nine scored the lowest difference between the manual schedule and the optimized one objective function values. Input used by that user showed that criteria number four was given the maximum degree of importance with respect to the other assigned goals, moreover, that goal was assigned a value that is already actualized – in other words cannot be breached – referring to the data of the available sections that was retrieved for that case, there was no courses which exam

A user can approach the optimal objective function value by giving the highest percentage of penalization to something that is already – or can be easily – actualized. Even if the system could satisfy the remaining goals, the user score will still be so close to the system one, because – in this case – these goals account for a relatively very small penalty.

The standard deviation and the arithmetic mean measure two different characteristics of a set of data. The arithmetic mean measures where the data is centered whereas the standard deviation measures how spread out the data is. Both values were calculated for the OFV reduction percentage data so as to evaluate the system efficiency.

A high (large) standard deviation indicates a wide range of scores or a great deal of variance. The greater the range of scores, the less representative the mean becomes.

A large standard deviation is usually indicated by comparing it to the mean. That is why precision is measured using the relative standard deviation which equal standard deviation divided by the mean. So that low values mean that data is precise.

The relative standard deviation can also be thought of as an index that gets smaller as the mean of the OFV's reduction percentages get bigger and their Dispersion gets smaller which is proper to measure the system efficiency.

CHAPTER 6: CONCLUSIONS AND RECOMMENDATIONS

6.1 Conclusions

According to the OFV estimation of both the prepared schedule and the DSSPS proposed one, and as a result of 25 tests of different students who used different insertions. DSSPS proposed schedule was improved by an average of 55.1% with respect to the manually prepared one with a standard deviation equals 0.1629 so that the relative standard deviation equals 0.2956. This means that about 68.2% of the readings deviate from the mean by no more than 29.5% of it.

The best improvement was 85.54% while the least one was 20.71%, but a gain this improvement depends on how complicated the problem is, as how many goals set by the user, for example, if the user defined only one goal– for example registering 15 hours – then, there will be no need to use the system at all, because such goal can easily be achieve. The more criteria get involved in the problem the more effective this system can be. Nevertheless, the previous values of both the average and standard deviation of the OFV reduction percentage data were a result of randomly tested students who were not obliged to maintain a certain level of complexity.

It took DSSPS almost a fraction of a second to calculate its optimized schedule for all of the cases while the tested students took from 15 to 30 minutes to prepare the manual schedule although they were in a rush, however in fact, this manual preparation usually takes much more time especially for students at the first two levels who may take days to reach a final registration.

DSSPS met a wide acceptance among the students. Importance survey results show that 72% of the students stated that this system is very important while 24% stated that it is important.

6.2 Limitations

a. The software was built on the assumption that multiple sections of the same course are held at the same timing. This is usually the case in universities that adopt the credit hours system. However, if it was not, then, a new searching mechanism should be developed to handle the task of constructing both hard and soft constraints that relate to timing conflicts issues.
b. For a student, desired courses, undesired courses, desired lecturers or undesired lecturers will not exceed eight per semester. Although, it is unlikely for any of the previously mentioned criteria to exceed eight, this limit can be extended easily through a few modifications of the software code and interface.
c. The software does not support more than one desired empty period within the same day, however, again this problem can be solved with a few programmatic work besides a slight modification to the interface.
d. This system is targeting the students, so, it assumes that type of the classes themselves, their timing, their lecturers and final exams are already set by the university so that this information will be treated by the system as fixed input. The system did not take into account lecturers or any other university employees preferences, So, a preliminary process that is similar to this one could be developed to assign classes, their timing and final exams in a way that form a compromise solution between all stakeholders.

e. The outcome of this process will be an input for the system developed here.
f. To use the system a computer – desktop or laptop – should be available that runs windows XP, vista or seven, furthermore, the software should be installed on it in case it was not embedded in the university website as a student service. This will usually be the case to transport calculations effort to the clients PC's. This issue raised the need for transforming this effort towards mobiles and portable devises.

6.3 Recommendations

6.3.1 Developing the software interface

All the work accomplished through this study including design of the system, design of the model, coding the generation process, the AHP part, the testing part, the translation process, designing the database and the information feeding mechanism, made it almost impossible to continue designing a fancy interface.

Although the interface already built through this study is quite suitable, it was mainly designed to test the system and not for a final end-user application.

The interface could be redesigned so that everything is categorized, obvious and easy to understand, it may include a variety of languages which are interchangeable, also, other helpful features can be added, for example, the possibility of saving the multiple scenarios created. This friendly interface could be the topic of a whole new study.

6.3.2 An easier and more efficient prioritizing method

During the process of testing the software, it was obvious that the students face a difficulty understanding the pairwise comparison method and how could it be applied through the relationship diagram half matrix. Even students from the engineering college failed to do it right.

On the other hand -as stated in chapter 5-, the user being unsatisfied with the resulting optimized schedule is usually a result of an inaccurate determination of his priorities. This happens because students fail to Realizes the difference between the units at which each criterion is penalized (described in chapter four).

Thus, a creative method to replace this traditional and rather complicated process may be developed. This method may clearly illustrate the difference between the various criteria and help reach an accurate prioritizing exactly the way meant by the user.

6.3.3 Targeting Lecturers

The same idea of this system can be used to develop anther one that targets lecturers. This system may be used for lectures scheduling, halls and lecturers allocation at the very beginning, however, other criteria should be defined, the ones that interest lecturers. Later on, Output of this system should be the input for the one developed here.

Moreover, there will be no database representing the available classes since they are still to be determined, instead, study plans, the number of students, the number of halls and labs will represent the initial input for the models generator.

This system will be used by the management. It will consider all lecturers preferences spontaneously and reach a compromise solution.

6.3.4 Moving to cloud computing

Cloud computing relies on sharing of resources to achieve coherence and economies of scale similar to a utility (like the electricity grid) over a network [40]. The cloud also focuses on maximizing the effectiveness of the shared resources.

Proponents claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of infrastructure [41].

In marketing, cloud computing is mostly used to sell hosted services in the sense of Application Service Provisioning that run client server software on a remote location. Such services are given popular acronyms like 'SaaS' (Software as a Service) and 'PaaS' (Platform as a Service). End users access cloud-based applications through a web browser or a light–weight desktop or mobile application while the business software and user's data are stored on servers at a remote location.

Consequently, developing a mobile application that emulates the function of the computer application built in this study would be of a great use especially in places where electricity supply is discontinuous, besides, a student will not necessarily always has access to a computer.

6.3.5 Implementation Plan

The Implementation Plan describes how the system will be deployed, installed and transitioned into an operational system. The plan contains an overview of the system, a brief description of the major tasks involved in the implementation, the overall resources needed to support the implementation effort (such as hardware, software. facilities, materials, and personnel) Including costs estimations.

Bibliography

[1] Chubb, D. W. J. (1984). “Knowledge Engineering Problems during Expert System Development.”

[2] A. M. Wittenstein and T. Sharma (2002). "FROSH2: An expert system for freshman advisement", Proceeding of the National Conference on Undergraduate Research (NCUR), University of Wisconsin, Whitewater, Wisconsin, USA, April 25-27.

[3] Marques, O., X. Ding, et al. (2001). “Design and development of a Web-based academic advising system.”

[4] Al Ahmar, M. A. (2011). “A Prototype Student Advising Expert System Supported with an Object-Oriented Database”.

[5] Kathryn Nobles (2007). “Academic Virtual Advisor”.

[6] Timmreck, E. M. (1968). “ADVISER - a program which advises students on courses.”

[7] Murray, W. S. and L. A. LeBlanc (1995). “A Decision Support System for Academic Advising.”

[8] Andres Scharifker (2010). “Virtual Academic Advisory: A solution using Integer Linear Optimization”

[9]. Raubinger, F. M., Rowe, H. G., Piper, D. L., and West, C. K. “The Development of Secondary Education. Old Tappan, N.J.: Macmillan, 1969.

[10]. John Harris, 2002. “BRIEF HISTORY OF AMERICAN ACADEMIC CREDIT SYSTEM: A Recipe for Incoherence in Student Learning”

[11]. “Credit Systems and Learning Outcomes in ASEM Member Countries” ASEM Seminar in Berlin, April 15—16, 2010

[12]. Ashford, Brenda (AACRAO). “2000-2001 Academic Calendars Study: Analytical Profiles of Calendar Use and Conversions”.

[13] CHO, K. (2003) “Multi Criteria Decision Methods: An Attempt to Evaluate and Unify” Mathematical and Computer Modeling, Vol. 37, (2003), pp 1099-1119

[14] Lootsma, F. (1999) “Multi-Criteria Decision Analysis via Ratio and Difference Judgment” Applied Optimization, Vol 29, Kluwer Academic Publishers, London

[15] Haarstrick, A., lazarevska, A. (2009). Multi-criteria decision making MCDM – a conceptual approach to optimal landfill monitoring.

[16] Dyer, R. and Forman, E. (1992) “Group decision support with the Analytic Hierarchy Process” Decision Support Systems, Vol. (8), (1992), pp 99-124 , North-Holland

[17] Bahurmoz, A. (2006) “The Analytic Hierarchy Process: A Methodology for Win-Win Management”, JKAU: Econ. & Adm., Vol. 20, No. 1, pp: 36-16

[18] Rifai, A. K. (1994). “A note on the structure of the goal programming model: Assessment and Evaluation” International Journal of Operations and Production Management, Vol. (16), pp 40–49.

[19] Romero, C. (1991) “Handbook of critical issues in goal programming” Oxford: Pergamon Press.

[20] Vencheh, A., Aghajani, M., (2010) “Designing a Production Programming Model with Multiple Objectives in Textile Industry” Australian Journal of Basic and Applied Sciences, Vol. 4, No. 9, pp 4390-4399.

[21] Tamiz, M., Jones, D., and El-Darzi, E., (1995) “A review of Goal programming and its applications”, Annals of Operations research, Vol. 58, No. 1, pp 39-53

[22] Wise, K. and Perushek, D. (2000), “Goal Programming as a Solution Technique”, Library Publications and Other Works. University of Tennessee, Knoxville, Available: http://trace.tennessee.edu/utk_libfpubs/25, (Accessed: 2011, June 22)

[23] Bertolini, M. and Bevilacqua, M. (2007) “A combined goal programming—AHP approach to maintenance selection problem” Reliability Engineering and System Safety, Vol. 91, (2006), pp 839–848

[24] Triantaphyllou, E., and Mann, S., (1990) "An Evaluation of the Eigen value Approach for Determining the Membership Values in Fuzzy Sets", Fuzzy Sets and Systems, Vol. 35, No. 3, pp. 295-301

[25] McGeehan, T., (1978) “Information service planning and evaluation: a goal programming approach”, (Doctoral Dissertation, Rutgers University)

[26] Ho, W., (2008) “Decision Support Integrated analytic hierarchy process and its applications – A literature review”, European Journal of Operational Research, Vol. 186, (2008), pp 211–228

[27] Druzdzel, M. J. and R. R. Flynn (1999). Decision Support Systems. Encyclopedia of Library and Information Science. A. Kent, Marcel Dekker, Inc.

[28] Alter, S. L. (1980). Decision support systems : current practice and continuing challenges.

Reading, Mass., Addison-Wesley Pub.

[29] Finlay, P. N. (1994). Introducing decision support systems. Oxford, UK Cambridge, Mass., NCC Blackwell; Blackwell Publishers.

[30] Turban, E. (1995). Decision support and expert systems : management support systems.

Englewood Cliffs, N.J., Prentice Hall.

[31] Keen, P. G. W. and M. S. Scott Morton (1978). Decision support systems : an organizational

perspective. Reading, Mass., Addison-Wesley Pub. Co.

[32] Sprague, R. H. and E. D. Carlson (1982). Building effective decision support systems.

Englewood Cliffs, N.J., Prentice-Hall.

[33] Ralph H. Sprague, Jr (1980). “A Framework for the Development of Decision Support Systems”

[34] Hättenschwiler, P. (1999). New user-friendly concept of decision support. Good decisions in business, politics and society. Zurich vdf, Hochschulverlag AG: 189-208.

[35] Power, D. J. (2002). Decision support systems: concepts and resources for managers. Westport, Conn., Quorum Books.

[36] Stanhope, P. (2002). Get in the Groove: building tools and peer-to-peer solutions with the Groove platform. New York, Hungry Minds

[37] Gachet, A. (2004). Building Model-Driven Decision Support Systems with Dicodess. Zurich, VDF.

[38] hil61217_ch07_supplement (2004).

[39] Forman, E. and Selly, M. (2002) "Decision By Objectives (How to convince others that you are right), World Scientific Pub Co Inc.

[40] "The NIST Definition of Cloud Computing". National Institute of Standards and Technology. Retrieved 24 July 2011.

[41] "What is Cloud Computing?". Amazon Web Services. 2013-3-19. Retrieved 2013-3-20.

Appendices

Appendix A: A Form Designed To Facilitate Testing Of A Decision Support System For Higher Education Student Preferences–Based Scheduling.

بسم الله الرحمن الرحيم

Islamic University Of Gaza

Deanery Of Higher Studies

Faculty Of Commerce

Department Of Management

A Form Designed To Facilitate Testing Of A Decision Support System

For Higher Education Student Preferences-Based Scheduling

Dear Student:

This form aims to test an optimization-based decision support system for higher education student scheduling process. Optimization done is mainly for the student personal preferences regarding his/her study schedule. These preferences represent the various characteristics of a study schedule such as the number of credit hours, the courses, the lecturers, lectures and final exams timing..etc.

This form aims to collect the required information about a group of students that enable the DSS creator to test the system by himself, since it is still too difficult to be used directly by students.

This form includes three parts, the first part is about a student personal desires. This part includes ten criteria which represent – from the author point of view – the most commonly considered. The student is supposed to define goals regarding the group of criteria that matter to him. At the end of this part, the student will be asked to suggest other criteria that may also be important.

The second part of this form is designed to define a student priorities for the goals he already defined in part one. This prioritizing process is conducted using the pairwise comparison method. (there is an example to clarify this process).

The third part is designed to perform a manual scheduling process in order to compare both schedules, the one prepared by the student manually and the one suggested by the system.

Note: the process of filling this form should be done assuming that the available classes is represented by the table exists in the last page of this form. (even though these classes are not really available for you at the moment)
Data collected will remain secret and will only be used by the researcher for scientific purposes.

All thanks and appreciation.

Researcher

Eng. Ahmad F. Abu Libda

Name:	**Number:**	**Major:**	**Level:**

First: Desired Setting: (fill only what matter you)

1- **Number of courses of each type:**
Departmental (<,>,=) ☐
College (<,>,=) ☐
University (<,>,=) ☐

2- **Desired number of credit hours (<,>,=)** ☐
3- **Least number of days to separate between final exams** ☐
4- **Furthest date of final exams …………………………..**

5. The desired courses (with or without a preferred lecturer)

	A desired course	A preferred lecturer
1		
2		
3		
4		
5		
6		
7		
8		

6. Number of days to precede a course final exam.

	A desired course	No of days to precede the final exam
1		
2		
3		
4		
5		
6		
7		
8		

7. Undesired courses

1	
2	
3	
4	
5	
6	
7	
8	

8. Non-preferred lecturers

1	
2	
3	
4	
5	
6	
7	
8	

9- Certain days within the week the student wishes to empty from lectures

Saturday ☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐

10- Certain periods throughout the week the student wishes to empty from lectures

Saturday [-] Sunday [-] Monday [-] Tuesday [-] Wednesday [-] Thursday [-]

Are there any other criteria that may interest you? ……………………………………………………………………………………………

Second: prioritizing. (Using pairwise comparisons)

Using numbers from 1 to 9 and ratios from ½ to 1/9 set relative importance between the various goals.

An example:

Criteria	**A**	**B**	**C**
A	▨	3	1
B	▨	▨	1/3
C	▨	▨	▨

- The number "3" means that criterion A is 3 times more important than criteria B.
- The ratio "1/3" means that criterion C is 3 times more important than criteria B.
- The number "1" means that both criteria A and C have the same importance.

Note:

- Fill only the white half of the matrix.
- Do not set comparisons between criteria that do not interest you.
- To leave a square empty is the same as assigning "1" to it. It means that the corresponding criteria have the same importance.

Main Criteria	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1- Number of courses of each type	▨									
2- Desired number of credit hours	▨	▨								
3- Least number of days to separate between final exams	▨	▨	▨							
4- Furthest date of final exams	▨	▨	▨	▨						
5- The desired courses (with or without a preferred lecturer)	▨	▨	▨	▨	▨					
6- Number of days to precede a course final exam.	▨	▨	▨	▨	▨	▨				
7- Undesired courses	▨	▨	▨	▨	▨	▨	▨			
8- Non-preferred lecturers	▨	▨	▨	▨	▨	▨	▨	▨		
9- Desired empty days	▨	▨	▨	▨	▨	▨	▨	▨	▨	
10- Desired empty periods	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Third: Manual Scheduling.

In order to test the software, a schedule should be constructed manually by the user as if he/she going to do it on his page at the university site. This schedule as well as the one suggested by the system will be evaluated.

For this evaluation to be fair, the user should take into account all the goals and priorities he has already defined in this form, but more importantly, he should avoid breaching registration regulations such as timing conflicts, minimum and maximum allowed number of credit hours…etc.

Use the available courses table in last page.

Record the time needed to construct the schedule:………………………….

Course number	Lecturer name	Section number	Lecture ending time	Lecture starting time	Lectures days (SNMTWH)	Final exam date	Final exam ending time	Final exam starting time	Course Name

What do you think of an optimization-based decision support system that tries to optimize achievement of the student personal desires regarding his/her schedule. This DSS will do the following:

1- Choose a group of sections that best satisfy the personal desires of a student represented in the previously mentioned ten criteria taking into account importance of each one.
2- This optimization process will avoid breaching any of the registration regulations; thus, introduce a schedule that is optimum, fast and feasible.

This DSS:

Very important ☐ Important ☐ Moderately important ☐ little important ☐ not important ☐

Do you have suggestions or comments?

………

………

The Available Sections Table

lecturer	End of final exam	Start of final exam	Lecture date	Lecture end	Lecture start	Lectures days	Sectio no	# of hours	Course Name	Course number
إبراهيم قدورة	1100	900	13/05/2013	1400	1200	W	114	1	فيزياء عامةعملية (ب)	PHYSB1102
محمد القريناوي	1100	900	13/05/2013	1000	800	W	112	1	فيزياء عامةعملية (ب)	PHYSB1102
معين عبيد	1100	900	13/05/2013	1400	1200	T	108	1	فيزياء عامةعملية (ب)	PHYSB1102
معين عبيد	1100	900	13/05/2013	1400	1200	T	108	1	فيزياء عامةعملية (ب)	PHYSB1102
حاتم الغمري	1100	900	13/05/2013	1000	800	M	106	1	فيزياء عامةعملية (ب)	PHYSB1102
إسلام رضوان	1100	900	13/05/2013	1000	800	M	106	1	فيزياء عامةعملية (ب)	PHYSB1102
إسلام رضوان	1100	900	13/05/2013	1200	1000	S	102	1	فيزياء عامةعملية (ب)	PHYSB1102
إبراهيم قدورة	1100	900	13/05/2013	1200	1000	S	102	1	فيزياء عامةعملية (ب)	PHYSB1102
حسين داوود	1100	900	22/05/2013	1100	930	NT	101	3	فيزياء عامة (ب)	PHYSB1301
سفيان تايه	1100	900	22/05/2013	1200	1100	SMW	102	3	فيزياء عامة (ب)	PHYSB1301
صالح النعامي	1630	1430	23/05/2013	1400	1300	SW	103	2	دراسات فلسطينية	POLS 3220
هاني البسوس	1630	1430	23/05/2013	1330	1230	NT	102	2	دراسات فلسطينية	POLS 3220
هاني البسوس	1630	1430	23/05/2013	1200	1100	SW	101	2	دراسات فلسطينية	POLS 3220
ناهض الشرفا	1400	1200	28/05/2013	1530	1400	NT	101	3	الآلات الكهربائية	EELE 3351
محمد خفاجة	1630	1430	20/05/2013	1100	1000	SW	101	2	مباديء الإقتصاد الإسلامي	ECON 4203
محمد خفاجة	1630	1430	20/05/2013	1100	1000	NT	102	2	مباديء الإقتصاد الإسلامي	ECON 4203
براء ريان	1630	1430	18/05/2013	1200	1100	SW	101	2	دراسات فى الحديث الشريف	HADT 4204
محمد المظلوم	1630	1430	18/05/2013	1500	1400	NT	103	2	دراسات فى الحديث الشريف	HADT 4204
رأفت نصار	1630	1430	18/05/2013	1400	1300	SW	102	2	دراسات فى الحديث الشريف	HADT 4204
صبحي اليازجي	1530	1430	27/05/2013	1100	1000	M	102	1	قرآن كريم (4)جزء الذاريات	HADTD2100
عبدالسلام اللوح	1530	1430	27/05/2013	1200	1100	M	101	1	قرآن كريم (4)جزء الذاريات	HADTD2100
صبحي اليازجي	1530	1430	27/05/2013	1000	900	M	103	1	قرآن كريم (4)جزء الذاريات	HADTD2100
أحمد أبولبدة	1400	1200	25/05/2013	1100	930	NT	101	3	عمليات التصنيع (1)	EIND 3303
جمال الزبدة	1400	1200	21/05/2013	1100	1000	SMW	101	3	ديناميكا حرارية 1	EMEC 3308
جمال الزبدة	1400	1200	20/05/2013	1300	1200	SMW	101	3	تصميم الآلات 1	EMEC 3306
جمعة العايدي	1600	1400	18/05/2013	1400	1200	N	101	1	معمل العلوم الحرارية	EMEC 3111
جمعة العايدي	1400	1200	29/05/2013	1400	1300	SMW	101	3	ديناميكا حرارية 2	EMEC 3313

Appendix B: A Form Designed To Facilitate Testing Of A Decision Support System For Higher Education Student Preferences–Based Scheduling. (In Arabic)

بسم الله الرحمن الرحيم

الجامعة الإسلامية بغزة

عمادة الدراسات العليا

كلية التجارة

قسم إدارة الأعمال

استبيان بهدف اختبار نظام دعم قرار لعملية التسجيل الفصلي للطالب الجامعي

على أساس تفضيلات الطالب الشخصية

عزيزي الطالب:

يهدف هذا الاستبيان إلى اختبار نظام دعم قرار مبني على أساس تحقيق الأمثلية لعملية الجدولة التي يقوم بها الطالب في بداية كل فصل دراسي.
يعتمد هذا النظام بشكل أساسي في عملية تحقيق الأمثلية على رغبات الطالب الشخصية المتعلقة بالجدول الدراسي. تتلخص هذه الرغبات في المواصفات الكمية والكيفية للعبء الدراسي الفصلي كعدد الساعات الأكاديمية ونوعية المساقات و المحاضرين و مواعيد المحاضرات والاختبارات النهائية..الخ.

يهدف الاستبيان إلى جمع معلومات عن عينة عشوائية من الطلبة في مختلف جامعات قطاع غزة تمكن الباحث من اختبار النظام بنفسه نظرا لأن واجهة البرنامج و خطوات الاستخدام ما زالت معقدة.

الاستبيان مكون من ثلاث أجزاء, الجزء الأول يتعلق برغبات الطالب الشخصية بخصوص الجدول الدراسي الفصلي الذي يود أن يقوم بتجهيزه. ويتلخص هذا الجزء في 10 معايير والتي تعد – من وجهة نظر الباحث - أهم و أكثر المعايير شيوعا. يقوم الطالب بملء رغبات معينه بخصوص المعايير التي تهمه فقط, وفي نهاية هذا الجزء يمنح الطالب مجالا لاقتراح معايير أخرى قد تهمه.

الجزء الثاني من الاستبيان مخصص لتحديد الأهمية النسبية لكل هدف قام الطالب بتسجيله في الجزء الأول, ويتم ذلك عن طريق إجراء مقارنات زوجية بين مختلف المعايير التي تهم الطالب. وهناك مثال توضيحي لهذه العملية.

الجزء الثالث من الاستبيان معد لإجراء عملية جدولة يدوية, وذلك لإجراء مقارنة بين الجدول المعد بواسطة الطالب وبين الجدول المقترح بواسطة النظام.

ملاحظة مهمة: عملية ملئ الاستبيان يجب أن تتم بافتراض أن الشعب المتاحة لك حاليا متمثلة بالجدول الموجود بآخر صفحة.(وإن كانت المساقات الموجودة به لا تخصك).

جميع المعلومات التي سيتم جمعها في هذا الاستبيان ستبقى سرية ولن تستخدم سوى لأغراض بحثية بواسطة الباحث فقط.

و لكم جزيل الشكر والتقدير.

الباحث:

م. أحمد فايز أبولبدة

الاسم:	**الرقم الجامعي:**	**التخصص:**	**المستوى:**

أولا: تحديد الرغبات: افترض أن الشعب المتاحة لك متمثلة بالجدول الموجود بآخر صفحة (**ملاحظة:** قم بملء المعايير التي تهمك فقط)

1. **عدد المساقات المرغوب من كل نوع (تخصص – كلية – جامعة)**

تخصص	**(= , < , >)**	☐
كلية	**(= , < , >)**	☐
جامعة	**(= , < , >)**	☐

2. **مجال عدد الساعات المرغوب (= , < , >)** ☐
3. **أقل عدد من الأيام التي تفصل بين الامتحانات النهائية** ☐
4. **تاريخ آخر امتحان نهائي: ....**

5.مساقات معينة مرغوبة (مع تحديد محاضر مرغوب أو بدون).

	المساق المرغوب	المحاضر المرغوب
1		
2		
3		
4		
5		
6		
7		
8		

6. عدد الأيام التي تسبق الامتحان النهائي لمساق معين.

	المساق المرغوب	الفترة التي تسبق الامتحان النهائي
1		
2		
3		
4		
5		
6		
7		
8		

7.مساقات معينة غير مرغوبة.

	المساق غير المرغوب
1	
2	
3	
4	
5	
6	
7	
8	

8. المحاضرين غير المرغوبين.

	المحاضر غير المرغوب
1	
2	
3	
4	
5	
6	
7	
8	

9. أيام معينة خلال الأسبوع يرغب بتفريغها (ضع علامة X في مربع اليوم (أو الأيام) الذي ترغب بتفريغه)

السبت ☐ الأحد ☐ الاثنين ☐ الثلاثاء ☐ الأربعاء ☐ الخميس ☐

10. أوقات معينة خلال الأسبوع يرغب بتفريغها (حدد فترة زمنية – من إلى - في مربعات الأيام التي تعنيك)

السبت	الأحد	الاثنين	الثلاثاء	الأربعاء	الخميس
-	-	-	-	-	-

هل هناك معايير أخرى تهمك أثناء قيامك بعملية التسجيل الفصلي؟ ....

ثانيا: تحديد الأولويات (باستخدام المقارنات الزوجية):

تعتمد هذه الطريقة استخدام الأرقام من 1 إلى 9 بالإضافة إلى النسب من ½ إلى 9/1 لتحديد الأهمية النسبية بين مختلف المعايير الداخلة كالتالي:

مثال توضيحي:

المعايير	A	B	C
A	▨	3	1
B	▨	▨	3/1
C	▨	▨	▨

- الرقم "3" يعني أن المعيار "A" أهم من المعيار "B" بثلاث مرات.
- النسبة "3/1" تعني أن المعيار "C" أهم من المعيار "B" بثلاث مرات.
- الرقم "1" يعني أن لكل من المعيار "A" والمعيار "C" نفس الأهمية.

ملاحظة:

- يقوم الطالب بملء الشق الأيسر (الغير مظلل) فقط.
- لا يجب إجراء مقارنات لمعايير لا تهمك (يقصد بها تلك المعايير التي لم تقم بتحديد أهداف معينة بخصوصها في الصفحة الأولى).
- ترك المربع فارغا يعني أن لكل من المعيارين نفس الأهمية.

المعايير	1	2	3	4	5	6	7	8	9	10
1. عدد المساقات المرغوب من كل نوع (تخصص – كلية – جامعة)	▨									
2. مجال عدد الساعات المرغوب	▨	▨								
3. عدد الأيام التي تفصل بين الامتحانات النهائية	▨	▨	▨							
4. تاريخ آخر امتحان نهائي	▨	▨	▨	▨						
5. مساقات معينة مرغوبة(مع تحديد محاضر مرغوب أو بدون)	▨	▨	▨	▨	▨					
6. عدد الأيام التي تسبق الامتحان النهائي لمساق معين	▨	▨	▨	▨	▨	▨				
7. مساقات معينة غير مرغوبة	▨	▨	▨	▨	▨	▨	▨			
8. محاضرين غير مرغوبين	▨	▨	▨	▨	▨	▨	▨	▨		
9. أيام معينة خلال الأسبوع يرغب بتفريغها	▨	▨	▨	▨	▨	▨	▨	▨	▨	
10. أوقات معينة خلال الأسبوع يرغب بتفريغها	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

ثالثا: الجدولة اليدوية: بهدف اختبار البرنامج يرجى تكوين جدول دراسي فصلي يدويا كما لو كنت ستقوم بهذه العملية على صفحتك بموقع الجامعة طبقا لما هو متوفر في جدول الشعب المتاحة (الموجود بآخر صفحة) وطبقا لرغباتك التي قمت بتسجيلها و مدى أهمية كل منها. **مع مراعاة تجنب التعارضات المتعلقة بمواعيد المحاضرات و الامتحانات النهائية بالإضافة إلى تجنب تسجيل عدد ساعات أكبر من الحد الأعلى أو أقل من الحد الأدنى المسموح به في حالتك.**

- **يرجى تسجيل الوقت الذي استغرقته لتكوين الجدول : ....**

اسم المساق	وقت بدء الامتحان النهائي	وقت انتهاء الامتحان النهائي	تاريخ الامتحان	أيام انعقاد المحاضرات (SNMTWH)	وقت بدء المحاضرة	وقت انتهاء المحاضرة	رقم الشعبة	اسم المحاضر	رقم المساق

ما رأيك ببرنامج كمبيوتر لدعم قرار عملية تجهيز الجدول الفصلي, يعتمد على أساليب كمية لتحقيق الأمثلية في الوصول إلى مجموعة من الأهداف المتمثلة في رغبات الطالب الشخصية المتعلقة بالجدول الدراسي الفصلي. يقوم هذا البرنامج بتجهيز جدول دراسي خال من التعارضات خلال ثواني حسب الأهداف التي سيقوم الطالب بإدخالها مما سبق ذكره ليصل هذا البرنامج إلى أفضل جدول دراسي يحقق جميع هذه الأهداف إن أمكن أو أغلبها حسب ما هو مطروح من مساقات و شعب و حسب ما تمثله أهمية كل هدف للطالب. أي أنه يقوم بالتالي:

1- اختيار مجموعة الشعب التي تحقق بقدر الإمكان أهداف الطالب الشخصية التي قام بتحديدها, و المتمثلة في المعايير العشرة السابق ذكرها, آخذاً بالاعتبار أهمية كل هدف منها بالنسبة للطالب.

2- ما ذكر في النقطة الأولى سيتم بالتزامن مع تجنب خرق قوانين الجامعة للتسجيل المتمثلة في تعارضات المحاضرات و تعارضات الاختبارات النهائية أو الحد الأقصى و الأدنى من الساعات الأكاديمية المسموح بتسجيله بالإضافة إلى اعتبارات أخرى.

هذا البرنامج:

☐ **شديد الأهمية** ☐ **مهم** ☐ **متوسط الأهمية** ☐ **قليل الأهمية** ☐ **غير مهم**

هل لديك تعليق أو اقتراحات؟

....

....

....

جدول الشعب المطروحة

رقم المساق	اسم المساق	عدد الساعات	رقم الشعبة	أيام انعقاد المحاضرات	بداية المحاضرة	نهاية المحاضرة	تاريخ الامتحان النهائي	وقت بدء الامتحان النهائي	وقت انتهاء الامتحان النهائي	المحاضر
PHYSB1102	فيزياء عامةعملية (ب)	1	114	W	1200	1400	13/05/2013	900	1100	إبراهيم قدورة
PHYSB1102	فيزياء عامةعملية (ب)	1	112	W	800	1000	13/05/2013	900	1100	محمد القريناوي
PHYSB1102	فيزياء عامةعملية (ب)	1	108	T	1200	1400	13/05/2013	900	1100	معين عبيد
PHYSB1102	فيزياء عامةعملية (ب)	1	108	T	1200	1400	13/05/2013	900	1100	معين عبيد
PHYSB1102	فيزياء عامةعملية (ب)	1	106	M	800	1000	13/05/2013	900	1100	حاتم الغمري
PHYSB1102	فيزياء عامةعملية (ب)	1	106	M	800	1000	13/05/2013	900	1100	إسلام رضوان
PHYSB1102	فيزياء عامةعملية (ب)	1	102	S	1000	1200	13/05/2013	900	1100	إسلام رضوان
PHYSB1102	فيزياء عامةعملية (ب)	1	102	S	1000	1200	13/05/2013	900	1100	إبراهيم قدورة
PHYSB1301	فيزياء عامة (ب)	3	101	NT	930	1100	22/05/2013	900	1100	حسين داوود
PHYSB1301	فيزياء عامة (ب)	3	102	SMW	1100	1200	22/05/2013	900	1100	سفيان تايه
POLS 3220	دراسات فلسطينية	2	103	SW	1300	1400	23/05/2013	1430	1630	صالح النعامي
POLS 3220	دراسات فلسطينية	2	102	NT	1230	1330	23/05/2013	1430	1630	هاني البسوس
POLS 3220	دراسات فلسطينية	2	101	SW	1100	1200	23/05/2013	1430	1630	هاني البسوس
EELE 3351	الآلات الكهربائية	3	101	NT	1400	1530	28/05/2013	1200	1400	ناهض الشرفا
ECON 4203	مباديء الإقتصاد الإسلامي	2	101	SW	1000	1100	20/05/2013	1430	1630	محمد خفاجة
ECON 4203	مباديء الإقتصاد الإسلامي	2	102	NT	1000	1100	20/05/2013	1430	1630	محمد خفاجة
HADT 4204	دراسات فى الحديث الشريف	2	101	SW	1100	1200	18/05/2013	1430	1630	براء ريان
HADT 4204	دراسات فى الحديث الشريف	2	103	NT	1400	1500	18/05/2013	1430	1630	محمد المظلوم
HADT 4204	دراسات فى الحديث الشريف	2	102	SW	1300	1400	18/05/2013	1430	1630	رأفت نصار
HADTD2100	قرآن كريم (4)جزء الذاريات	1	102	M	1000	1100	27/05/2013	1430	1530	صبحي اليازجي
HADTD2100	قرآن كريم (4)جزء الذاريات	1	101	M	1100	1200	27/05/2013	1430	1530	عبدالسلام اللوح
HADTD2100	قرآن كريم (4)جزء الذاريات	1	103	M	900	1000	27/05/2013	1430	1530	صبحي اليازجي
EIND 3303	عمليات التصنيع (1)	3	101	NT	930	1100	25/05/2013	1200	1400	أحمد أبولبدة
EMEC 3308	ديناميكا حرارية 1	3	101	SMW	1000	1100	21/05/2013	1200	1400	جمال الزبدة
EMEC 3306	تصميم الآلات 1	3	101	SMW	1200	1300	20/05/2013	1200	1400	جمال الزبدة
EMEC 3111	معمل العلوم الحرارية	1	101	N	1200	1400	18/05/2013	1400	1600	جمعة العايدي
EMEC 3313	ديناميكا حرارية 2	3	101	SMW	1300	1400	29/05/2013	1200	1400	جمعة العايدي

Appendix C: The available classes query block written in SQL

```
SELECT
STUDENT_NO,
A.SUBJECT_NO,
A.SUBJECT_A_NAME,
B.SMTR_NO,
B.BRANCH_NO,
DAY,
TIMEFROM,
TIMETO,
EXAM_DATE,
EXAM_TIMEFROM,
EXAM_TIMETO,
SUBJECT_TEACHER_NAME(K.SMTR_NO,K.SUB_NO,K.BRANCH_NO) EMP_NAME
FROM REMAIN_STD_SUBJECT A, SUBJECT_ROOM_TIMES B ,SUB_TEST  S , sub_teacher k
WHERE A.SUB_NO=B.SUB_NO
AND A.SMTR_NO=B.SMTR_NO
AND A.SMTR_NO=20132
AND A.SUB_NO=S.SUB_NO
AND A.SMTR_NO=S.SMTR_NO
AND B.SUB_NO=K.SUB_NO
AND B.SMTR_NO=K.SMTR_NO
AND B.BRANCH_NO=K.BRANCH_NO
AND STUDENT_NO=120101013
AND SUBSTR(B.BRANCH_NO,1,1) <>2
AND SUBSTR(B.BRANCH_NO,2,1) <>5
```

Appendix D: Functions written in VB used to convert data enquired to the familiar software format.

Function used to convert days.

```
Function Dc(Dn As String) As String

Dc = ""
For n = 1 To Len(Dn)
Dc = Dc & ConvertDay(Mid(Dn, n, 1))
Next n

End Function

Function ConvertDay(S1 As String) As String

Select Case S1
Case "S"
ConvertDay = "1"
Case "N"
ConvertDay = "2"
Case "M"
ConvertDay = "3"
Case "T"
ConvertDay = "4"
Case "W"
ConvertDay = "5"

End Select

End Function
```

Function used to convert time.

```
Function Tc(Tn As String) As Double

If Mid(Tn, Len(Tn) - 1, 1) = "3" Then
Tc = ((Val(Tn) - 830) / 50) + 1
Else
Tc = ((Val(Tn) - 800) / 50)
End If

End Function
```

Code used to rename the fields

```
CurrentDb.TableDefs("divisions").Fields("xxx").Name = "s"
CurrentDb.TableDefs("divisions").Fields("xxx").Name = "e"
CurrentDb.TableDefs("divisions").Fields("xxx").Name = "lecturer"
CurrentDb.TableDefs("divisions").Fields("xxx").Name = "dno"
CurrentDb.TableDefs("courses").Fields("xxx").Name = "s"
CurrentDb.TableDefs("courses").Fields("xxx").Name = "e"
CurrentDb.TableDefs("courses").Fields("xxx").Name = "examd"
```