SOLVING THE THESIS DEFENSE TIMETABLE PROBLEM USING HEURISTIC METHODS

By

MAHA ALI MOHAMED ALBASHIR

INDEX NO. 114089

Supervisor

Dr. Ahmed Kheiri

A REPORT SUBMITTED TO

University of Khartoum

in partial fulfillment of the requirements for the degree of

B. Sc. (HONS) Electrical and Electronic Engineering

SOFTWARE ENGINEERING OF ELECTRONIC SYSTEMS

Faculty of Engineering

Department of Electrical and Electronic Engineering

October 2017
DECLARATION OF ORGINALITY

I declare this report entitled “Solving The Thesis Defense Timetabling Problem using Heuristic Methods” is my own work except as cited in references. The report has been not accepted for any degree and it is not being submitted currently in candidature for any degree or other reward.

Signature: __________________

Name: ______________________

Date: ________________________
ACKNOWLEDGEMENT

In the name of ALLAH, Most Gracious, Most Merciful.

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my remarkable supervisor Dr. Ahmed Kheiri because he was not just the supervisor that guide, support and help but also the brother, the father and the friend. Through his support and trust, I have had the opportunity to work on an interesting project and to learn a lot from his advice and guidance.

A special thanks goes to my team mate Elham because this project can not be completed without her effort and co-operation.

I am highly obliged in taking this opportunity to sincerely thanks to all the staff members of Electrical and Electronic department for their generous attitude and friendly behaviour.

Finally I apologize all other unnamed who help me in various ways to complet this project.
DEDICATION

This work is dedicated with love and affection to those who appreciate time

To my beloved mother

The hand that captures me after each fall, everything I am you helped me to be

To my dear father

For always being there for me

To my lovely sisters

For always helping me go through difficulties, you are the reason why I am here today

To my fiancé

For his patient and support

To my friends

For their endless support
ABSTRACT

This project aimed to solve the problem of timetabling specifically Thesis Defense Timetabling problem. For many decades solving timetabling problems and producing an optimal solution was considered a hard problem, since producing manual timetables requires a lot of time and effort. Solving thesis defense timetabling problem requires scheduling of group of projects into sessions and rooms and assigning a number of teachers to them as a committee according to certain constraints. This problem is new to the literature and a little work is done in the area of thesis defense timetabling problem.

For solving this problem, hyper-heuristic approach was used. Hyper-heuristics are considered as an improvement methodologies that control a set of low level heuristics during the search process. In this study, the performance of set of low level heuristics was evaluated along with three move acceptance methods to produce the better solution from all. The best solution resulted in considerable reduction of constraints violations. This hyper-heuristic has been tested using a real world data coming from University of Khartoum.
المستخلص

المشروع المقدم في هذه الأطروحة يهدف إلى إيجاد حل لمشكلة إنشاء الجداول وتحديداً إنشاء جداول لمناقشة مشاريع التخرج.

في سابق العهد ولعدم وجود معالات إنشاء الجداول تعد من أصعب العمليات الحسابية، فقد كانت هذه العملية تم útilًا وتتطلب الكثير من الوقت والجهد. عملية إنشاء جداول المناقشات تتطلب إيجاد لجنة مناسبة من الأساتذة وترتيبهم على المشاريع في فترة زمنية محددة وقاعات محددة بشرط إستيفاء بعض الشروط العملية في الحل. لم يتم تداول مسألة مشاريع التخرج وإيجاد بعض الحلول لها بصورة كبيرة في البحوث والمشاريع.

في هذا المشروع تم استخدام طريقة أساليب الإستدلال المفرط في الحل التي تعتبر طريقة من طرق تحسين الحل بتحكيمها في أساليب الاستدلال منخفضة المستوى خلال عملية البحث عن الحل الأفضل. تم حساب الأداء لكل من أساليب الاستدلال منخفضة المستوى بعد دمجها مع عملية تهتم بقبول الحل الناتج حسب بعض القيود. من خلال هذا المشروع لوحظ أن الحل الأخير لم يكن الأمثل على الإطلاق ولكن به ساعد كثيراً في استيفاء بعض القيود. استخدمت هذه الطريقة على بيانات حقيقية من جامعة الخرطوم.
# TABLE OF CONTENTS

DECLARATION OF ORGINALITY .................................................................................. ii

ACKNOWLEDGEMENT ................................................................................................. iii

DEDICATION ................................................................................................................ iv

ABSTRACT ...................................................................................................................... v

المستخلص ..................................................................................................................... vi

TABLE OF CONTENTS ................................................................................................... vii

LIST OF FIGURES .......................................................................................................... x

LIST OF TABLES ............................................................................................................. xii

LIST OF ABBREVIATIONS ............................................................................................. xiii

1. INTRODUCTION ....................................................................................................... 1
   1.1 Overview .............................................................................................................. 1
   1.2 Background ........................................................................................................ 1
   1.3 Problem Statement ............................................................................................ 1
   1.4 Objectives ......................................................................................................... 2
   1.5 Motivation ........................................................................................................ 2
   1.6 Thesis Organization ......................................................................................... 3

2. LITREATURE REVIEW .............................................................................................. 4
   2.1 Classical Solutions ............................................................................................ 4
   2.2 Artificial Intelligence ......................................................................................... 4
   2.3 Heuristics ......................................................................................................... 6
       2.3.1 Meta-heuristics ....................................................................................... 6
       2.3.2 Hyper-heuristics ................................................................................... 7
       2.3.3 Hyper-heuristics Types .......................................................................... 8
2.3.3.1 Classification according to learning .......................................................... 9
2.3.3.1.1 Hyper-heuristics without learning .......................................................... 9
2.3.3.1.2 Hyper-heuristics with learning ............................................................. 9
2.3.3.2 Classification according to the nature of the heuristic search space .... 9
2.3.3.2.1 Generative hyper-heuristics ............................................................... 9
2.3.3.2.2 Selective hyper-heuristics ................................................................. 10
2.3.3.3 Classification according to the nature of the low-level heuristics used
in the hyper-heuristic framework .................................................................. 10
2.3.3.3.1 Constructive hyper-heuristics ............................................................ 10
2.3.3.3.2 Perturbative hyper-heuristics ............................................................ 11

2.4 Hyper-heuristic Application: ........................................................................ 14
2.4.1 Traveling Salesman Problem (TSP) ......................................................... 14
2.4.2 Nurse scheduling problem (NSP) ............................................................ 15
2.4.3 Timetabling problems ............................................................................. 15

2.5 Classification of Educational Timetabling Problems .................................. 17
2.5.1 School Timetabling ................................................................................. 17
2.5.2 Course Timetabling .................................................................................. 17
2.5.3 Examination Timetabling ........................................................................ 18
2.5.3.1 TDTP .................................................................................................. 22

2.6 The International Timetabling Competition (ITC): .................................. 22

3. PROBLEM STATEMENT ................................................................................... 24
3.1 Introduction: .................................................................................................. 24
3.2 Problem Description .................................................................................... 24
3.3 Solving timetabling problem using classical methods .................................. 25
3.4 Constraints of the problem ................................................................. 26

4. METHODOLOGY .................................................................................. 28

4.1 Overview ............................................................................................. 28

4.2 Introduction .......................................................................................... 28

4.3 Problem Instances .............................................................................. 28

4.4 Data setup ............................................................................................ 30

4.5 Solution Representation: ..................................................................... 31

4.5.1 Sessions solution representation ..................................................... 31

4.5.2 Examiners solution representation ................................................. 31

4.6 Solution steps ....................................................................................... 32

4.7 Initialization: ....................................................................................... 33

4.7.1 Constant value initialization ............................................................. 33

4.7.2 Random initialization ........................................................................ 34

4.8 Low level heuristics ............................................................................ 35

4.8.1 Utilization of low level heuristics .................................................... 37

4.9 Evaluation method .............................................................................. 38

4.10 Hyper-heuristic .................................................................................. 39

4.10.1 Heuristic selection ......................................................................... 39

4.10.2 Move acceptance ............................................................................ 40

4.10.2.1 Accept all moves ....................................................................... 40

4.10.2.2 Accept only improving moves ................................................... 40

4.10.2.3 Accept Improving or equal moves ........................................... 41

5. RESULTS .............................................................................................. 42

5.1 Overview ............................................................................................. 42
5.2 Final results................................................................................................................. 42
5.3 Comparison between low level heuristics: ......................................................... 45
6. Conclusion ..................................................................................................................... 47
  6.1 Overview ..................................................................................................................... 47
  6.2 Summary of work ...................................................................................................... 47
  6.3 Future work ................................................................................................................. 47
References ....................................................................................................................... 48
APPENDIX A..................................................................................................................... A-1
APPENDIX B..................................................................................................................... B-1
APPENDIX C..................................................................................................................... C-1
APPENDIX D..................................................................................................................... D-1
LIST OF FIGURES

Figure 2-1: Hyper-heuristic Framework ................................................................. 8

Figure 2-2: A classification of hyper-heuristic approaches ..................................... 14

Figure 4-1: Flow chart for solution steps ............................................................... 32

Figure 4-2: Example of changing assignment randomly ....................................... 36

Figure 4-3: Example of un-assigning randomly ...................................................... 36

Figure 4-4: Example of changing assignment randomly ....................................... 37

Figure 4-5: Example of randomly swap ................................................................. 37

Figure 4-6: Selection hyper-heuristic ................................................................. 39

Figure 5-1: shows improvement of cost ................................................................. 44

Figure 5-2: shows the utilization for LLH1, LLH2 and LLH3 for sessions solution representation ................................................................. 45

Figure 5-3: shows the utilization for LLH1, LLH2 and LLH3 for examiners solution representation ................................................................. 46

Figure D-1: shows user interface ................................................................. 6-5
LIST OF TABLES

Table 2-1: Basic terminology used in timetabling [31] .......................................................... 16
Table 3-1: Time required to solve timetabling problem using Sequoia................................. 25
Table 3-2: Time required to solve timetabling problem using Sequoia................................. 26
Table 4-1: Data Instances summary ....................................................................................... 28
Table 4-2: Example of session solution representation ........................................................... 31
Table 4-3: Example of examiners solution representation ......................................................... 31
Table 4-4: Example of session solution representation after initialization............................ 33
Table 4-5: Example of examiners solution representation ......................................................... 34
Table 5-1: Results obtained from using random initialization with different heuristic.... 42
Table 5-2: Results obtained from using constant initialization with different heuristic... 43
Table 5-3: Cost improvement .................................................................................................. 44
Table B-1: Supervisors_Examiners sheet ................................................................................. 6-1
Table B-2: Projects .................................................................................................................. 6-1
Table B-3: rooms ..................................................................................................................... 6-2
Table B-4: Sample of suitability sheet ..................................................................................... 6-1
Table B-5: Availability sheet .................................................................................................... 6-1
Table C-1: Results sample ........................................................................................................ 6-1
# LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AI</strong></td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td><strong>ITC</strong></td>
<td>The International Timetabling Competition</td>
</tr>
<tr>
<td><strong>LLH</strong></td>
<td>Low Level Heuristic</td>
</tr>
<tr>
<td><strong>NSP</strong></td>
<td>Nurse Scheduling Problem</td>
</tr>
<tr>
<td><strong>TDTP</strong></td>
<td>Thesis Defense Timetabling Problem</td>
</tr>
<tr>
<td><strong>TSP</strong></td>
<td>Traveling Salesman Problem</td>
</tr>
<tr>
<td><strong>U of K</strong></td>
<td>University of Khartoum</td>
</tr>
</tbody>
</table>
CHAPTER 1

INTRODUCTION

1.1 Overview

This chapter will discuss about the overall of the project. It has five sections. The first section will discuss the background of the project, then the problem statement will be explained in the second section. The third section will contain the objectives of the project. Motivation will be mentioned in the fourth section while the last section will explain the thesis organization.

1.2 Background

Timetabling problems are one of the most computationally difficult problems in scheduling. Solving a real world timetabling problems manually often requires a lot of time, resources and effort. Automation of the process of timetabling generation save a lot of time and effort and also improve the quality of timetables whereas eliminating the human factor result in avoiding a lot of errors and conflicts. Timetabling problems are of several types including educational timetabling, hospital timetabling and transportation timetabling. Educational timetabling is the most studied among all the timetabling problems. Timetabling problems have attracted the attention of the scientific community from a number of disciplines (including Operations Research and Artificial Intelligence) for about forty years and over the last decade or so there has been increased interest in the field [1]. A lot of research has been invested in this area over the years trying to automate the creation of timetables. Generally in solving such problems an initial solution is constructed first then an improvement is done to this solution to come up with new solution then some criteria decide whether to accept this new solution it or not.

1.3 Problem Statement

Educational timetabling can be categorized into examination and course timetabling problems. We consider the problem Thesis Defense Timetabling Problem (TDTP) Using
Heuristic Methods. It is concerned with assigning graduation projects presentations with some number of examiners to a specific number of timeslots and rooms and with respect to a given constrains. This constrains can be categorize into two categories, hard constrains and soft constrains. Hard constrains must not be violated and the solution is considered feasible if it satisfy all the hard constrains for the problem. Examples of hard constrains could be assigning two project presentations into one room at the same time and assigning one examiner into two projects presentations in the same timeslot. Soft constrains, on the other hand, should be satisfied as hard as possible but if they are not satisfied the solution still remains feasible as far as the hard constrains are satisfied. A weighted Penalty value is calculated whenever some soft constrains are violated and the objective is to minimize it. Soft constrains used to evaluate the optimization and the quality of the solution. Examples of soft constrains could be assigning an examiner to a project presentation while he is not interested in and assigning an examiner into all timeslots. A real world data from University of Khartoum has been used. Hyper heuristics are to be used in this study. A hyper-heuristic is used to define a high-level heuristic that controls low-level heuristics [2]. Level heuristics are chosen according the problem statement.

1.4 Objectives

The objectives of this study are as follows:

1. Specifying the hard and the soft constraints that are defining in the problem.
2. Implementing solution representation and initialization method and evaluation method for the problem.
3. Implementing hyper heuristic methods that satisfy all the hard constrains and produce feasible timetables for University Of Khartoum.

1.5 Motivation

Automation has become everywhere, almost in every single detail in life. Solving a timetabling problem manually is extremely difficult at some circumstance and requires a lot of time and effort. One single error in the timetabling process can affect the organization as a whole. Hence to handle the complexity of the manual creation of
timetables and to provide automated support hyper heuristics are used. The best solution has always been the goal which people require from solving computational optimization problems but using hyper heuristics aims to generalize to solution in order to be applied to many other problems. This facts was what motivate this study. Another thing that motived this work, was to learn new approaches. Hyper-heuristic is considered a new approach in solving such problems. Not so much work has been done in the area of using hyper-heuristic to solve optimization problems. Also, enhancing the team work and presentation skills was also a motivation for this work.

1.6 Thesis Organization

This thesis consists of five chapters organized in the following way:

Chapter 1 (Introduction): Contains the background, the problem statement, the objectives and the motivation.

Chapter 2 (Literature Review): Provides the literature review of timetabling problem and the methods that have been used to solve this problem.

Chapter 3 (Methodology): Describes the methodology used in this study.

Chapter 4 (Results and Discussion): Shows the result that is generated from the solution and discussing them.

Chapter 5 (Conclusion): Presents the overall conclusions of the work done, limitation and research directions for future work.
CHAPTER 2

LITREATURE REVIEW

This chapter provides details of the fundamental aspects of the timetabling problem. It describes in details some of the methods used to solve timetabling problems.

2.1 Classical Solutions

Solving timetabling problems using classical methods requires a lot of effort and time. Even the fastest computer in the world will need hundred years to find all the possible combinations of the timetable with the specific constraints. It will also need another time to determine which one is the optimal timetable. Researchers start to search for other solutions methods. Mathematicians start to use their methods to discard some possible combinations that are not suitable as a solution for the specific problem. This reduces the time but not so much. Using heuristics is the best solution that takes reducing time into consideration, so far.

2.2 Artificial Intelligence

Artificial Intelligence (AI) techniques try to simulate human behavior. They present a better, faster and accurate solution to an optimization problem than the existing conventional techniques. Artificial intelligence techniques generally make use of multiple solutions to obtain an optimal solution. Some of the peculiar properties of these techniques are:

1. These methods remember past findings.
2. They learn and adapt their performance.
3. These methods can plan their path forward.
4. These methods act intelligently according to human or social intelligence [3].

AI research community is quite active in the area of timetabling and scheduling and has developed a variety of approaches for solving such problems. They can be roughly divided into four types [1]:


4
• Sequential methods
These methods order events using domain heuristics and then assign the events sequentially into valid time periods (also called timeslots) so that no events in the period are in conflict with each other; events are most often ordered in such a way that events that are most difficult to schedule are assigned into timeslots first [4].

• Cluster methods
In this methods events are collected in clusters where any two events in a particular cluster do not conflict with each other; the main drawback of these approaches is that the clusters of events are formed and fixed at the beginning of the algorithm and that may result in a poor quality timetable [5].

• Constraint-based approaches
In these methods a timetabling problem is modelled as a set of variables (i.e., events) that have a finite domain to which values (i.e., resources such as time periods) have to be assigned to satisfy a number of constraints; a number of rules is defined for assigning resources to events and when no rule is applicable to the current partial solution a backtracking is performed until a solution is found that satisfies all constraints; as the satisfaction of all constraints may not be possible, algorithms are generally allowed to break some constraints in a controlled manner in order to produce a complete timetable([6], [7]).

• Meta-heuristic methods
Variety of meta-heuristic approaches such as simulated annealing, tabu search, evolutionary algorithms and hybrid approaches have been investigated for timetabling; meta-heuristic methods begin with one or more initial solutions and employ search strategies to find optimal solution, trying to avoid local optima in the process ([5], [8], [9], [10], [11]). Recently traditional combinatorial optimization solutions are not highly preferred to be used in solving timetabling problems because of their highly computational cost. Traditional solutions can produce good solution that solve the specific problem but they are sometimes not suitable to be used to solve large, high constrained problems. Most solutions start to use heuristic instead.
2.3 Heuristics

Heuristics are good and fast solution for optimization problems but not guaranteed to be the most optimal. Solving problems using heuristic solutions may yield to an approximated solution for the problem not like the classic solutions which always try to find the exact solution for the specific problem. The solution solved by using heuristics may not seems to be the best solution although it is still valuable because it does not require as much time as the best solution may need. Heuristics are used when the information about the problem is incomplete or when the time required to produce the solution is a critical issue and must be minimized to the lowest degree. Some types of heuristics are:

2.3.1 Meta-heuristics

Turing’s pioneer work in heuristic search has inspired many generations of research in heuristic algorithms [12]. In the last two decades, meta-heuristic algorithms have attracted strong attention in scientific communities with significant developments, especially in areas concerning swarm intelligence based algorithms [12]. Turing called his search method heuristic search, as it could be expected to work most of time, but there was no guarantee to find the true or correct solution, yet it was a tremendous success [13]. Glover and Laguna 1997 defined meta-heuristic as: “a meta-heuristic refers to a master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality”. The heuristics guided by such a meta-strategy may be high level procedures or may embody nothing more than a description of available moves for transforming one solution into another, together with an associated evaluation rule. Meta-heuristics are heuristics that aim to find, generate or select low level heuristics that lead to a good solution for optimization problems. They are used to designate a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. For the past year meta-heuristics have been used to solve a lot of combinatorial optimization problems. New meta-heuristics were also developed to solve different kinds of problems and problem specific systems were produced. Meta-heuristic methods begin with one or
more initial solutions and employ search strategies that try to avoid local optima. All of these search algorithms can produce high quality solutions but often have a considerable computational cost [1].

2.3.2 Hyper-heuristics

Although the use of meta-heuristics have solved a lot of problems but the limitation that encouraged the researcher to search for another solution is that meta-heuristics require extensive knowledge about the problem because it operate directly on a search space of solutions. Also one solution may be impossible to be used for other problem domains without modification. This called for the emergence of another approach, called hyper-heuristics.

Hyper-heuristics can be defined as heuristics which choose heuristics, as opposed to searching for a solution directly [14]. The basic idea of hyper-heuristics has been around since the 1960s when Fisher and Thompson presented an algorithm which combines local job shop scheduling rules using a probabilistic learning( [15], [16]). The learning technique was used to choose one of two heuristics which assign jobs to machines and therefore it can be classified as a hyper heuristic. Hyper-heuristic research is concerned with building systems which can automatically guide and adapt a search according to the structure of the problem in hand. This can be achieved through the exploitation of the structure of a problem, and the creation of new heuristics for that problem, or intelligently choosing from a set of pre-defined heuristics [17]. Hyper-heuristics aims to raise the level of generality at which optimization systems can operate. Always there is a general system that automatically analyze the problem and creates or chooses predefined heuristics making the solution applicable for other problems. However, raising the generality of such a system will lead to, on the other hand, reducing the quality of the solution. Hyper-heuristic research aims to automate the heuristic design process to produce “good-enough soon enough cheap-enough” solutions for organizations with similar optimization needs [18]. The broad aim is to design an algorithm for solving a range of problems or instances of a problem that is fast, reasonably comprehensible, trustable in terms of quality and reliable in terms of its performance across that range of problems [14]. Figure 2.1 presents a diagram of a general Hyper-heuristic framework. It shows that hyper-heuristic
can only access the non-domain data due to the presence of the domain barrier [17]. Therefore, the hyper-heuristic does not require any knowledge of the problem being solved. It only deals with the low level heuristics that have access to the domain data and can directly act on the problem. The evaluation function then evaluates the solution and the results are passed to the hyper-heuristic which will take decisions accordingly [17].

![Hyper-Heuristic Framework](image)

**Figure 2-1: Hyper-heuristic Framework [19]**

### 2.3.3 Hyper-heuristics Types

Hyper-heuristics can be classify into different classes according to different criteria.
2.3.3.1 Classification according to learning

2.3.3.1.1 Hyper-heuristics without learning

The hyper-heuristics without learning include approaches that use several heuristics (neighborhood structures), but select the heuristics to call according to a predetermined sequence. Therefore, this category contains approaches such as variable neighborhood search [21].

2.3.3.1.2 Hyper-heuristics with learning

The hyper-heuristics with learning include methods that dynamically change the preference of each heuristic based on their historical performance, guided by some learning mechanism. As discussed in [20], hyper-heuristics can be further classified with respect to the learning mechanism employed, and a distinction is made between approaches which use a genetic algorithm, from those which use other mechanisms. Hyper-heuristics with learning can also be categorized into two types:

- Hyper-heuristics with online learning

In this type the learning mechanism takes place while the algorithm is solving an instance of a problem and the adaptation is done immediately. Using meta-heuristic over the search space and the use of reinforcement learning for heuristic selection are two examples of the online learning.

- Hyper-heuristics with offline learning

In this type some information are gathered from a set of training instances to generalize a set of rules for the solution. Genetic programming and learning classifier systems are two examples of the offline learning.

2.3.3.2 Classification according to the nature of the heuristic search space

2.3.3.2.1 Generative hyper-heuristics

It aims to create new heuristics from a set of components. The true challenge here falls in the creation of a new heuristic or a variation of a previously created heuristic, which may obtain the best possible result for an instance [17]. It always focuses on creating general solution that can be apply to a different instances of the same problem class. Therefore, if
the heuristic design process is automated, a computer system could produce a good quality heuristic for an instance in a practical amount of time [17]. This solution could be even better than the solution produced by the human created the heuristic.

2.3.3.2.2 Selective hyper-heuristics

It aims to choose heuristics from a set of predefined heuristics. This can be seen as a framework to automate the process of predefined heuristic choice and hybridization [17]. Testing different heuristics in a specific problem to get the desired solution is a time consuming process. Therefore, hyper-heuristics use intelligent systems that could decide the appropriate heuristics to be used to solve a certain problem. In this type the hyper-heuristic decides which heuristics to be applied to the specific problem according to its current state. The hyper-heuristic perform search over the heuristic space instead of searching over the solution space. After choosing a certain heuristic to be used, this heuristic acts on the problem and the solution is evaluated using an objective function. Finally, the hyper-heuristic uses this evaluation to decide whether to accept the current heuristic or switch to a different one [17]. A framework has been presented in ([2], [21] [22] [23] [24]) where the hyper-heuristic operates at a higher level of abstraction without using any domain knowledge. The hyper-heuristic has no knowledge about the problem being solved it can only access the low-level heuristics. Therefore, the hyper-heuristic can be re-used to solve other different problem by replacing the low-level heuristics and the objective function.

2.3.3.3 Classification according to the nature of the low-level heuristics used in the hyper-heuristic framework

2.3.3.3.1 Constructive hyper-heuristics

Constructive hyper-heuristics build a solution incrementally by adaptively selecting heuristics, from a pool of constructive heuristics, at different stages of the construction process [25]. It starts with an empty solution to intelligently select and use construction heuristics to get to the complete solution. The hyper-heuristic framework is provided with a set of preexisting (generally problem specific) construction heuristics, and the challenge
is to select the heuristic that is somehow the most suitable for the current problem state [26]. This process continues until the final solution is obtained.

2.3.3.3.2 Perturbative hyper-heuristics

They also called Local search hyper-heuristics. They start from a complete initial solution and iteratively select, from a set of neighborhood structures, appropriate heuristics to lead the search in a promising direction [25]. The goal of the perturbative hyper-heuristics is to improve the current existing solution. The hyper-heuristic framework is provided with a set of neighborhood structures and/or simple local searchers, and the goal is to iteratively select and apply them to the current complete solution [26]. This process continues until a stopping condition is met.

Hyper-heuristics consists of two phases: heuristic selection method and move acceptance.

1. Heuristic selection:

It is the process of selecting and applying a heuristic from a set of low level heuristics in order to solve a given problem. This chosen heuristic can be applied to other problem domain without any modification. There are many methods for heuristic selection such as:

1.1. Simple function:
   1.1.1. Simple Random:

   This method randomly selects low level heuristic at each decision point.

   1.1.2. Random Descent:

   This method randomly chooses low level heuristic and applies it to the solution as far as there is an improvement.

   1.1.3. Random Permutation:

   This method randomly generates a permutation of heuristics and applies one heuristic at a time in that order.

   1.1.4. Random Permutation Descent:
This method is based on the same RP strategy, however similar to random decent, but it applies the same heuristic repeatedly until there is no more improvement.

1.2. **Greedy Function:**

This method applies all the low level heuristic to generate many solutions then choose the best one.

1.3. **Choice Function:**

This method is an online learning heuristic selection method that scores each low level heuristic based on their utility value and selects the one with the highest score [27].

1.4. **Tabu Search:**

The idea of tabu search was proposed by Fred Glover (1986). Glover and Laguna (1997) define tabu search as: “a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality”. This method firstly give each heuristic a rank. For every improvement appears in the solution the rank of the heuristic is increased by a positive value yet there is a maximum value the rank cannot exceed. On the other hand if no improvement appears in the solution the rank is decreased by a negative value yet there is a minimum value the rank can not exceed. If worsening moves appears, the heuristic is also added to the tabu list. A maximum number of iterations a low level heuristic can stay in the tabu list is also determined by the tabu duration.

2. **Move acceptance:**

It is the process that determine whether to accept this generated solution or not. There are many methods for heuristic selection such as:

2.1. **Accept all moves:**

This method accepts all the low level heuristics used to produces solution.

2.2. **Accept only improving moves**

This method accepts only low level heuristics that produce a solution better than the current one.
2.3. Accepting Improving or Equal move

This method accepts only low level heuristics that lead to a better or equal solution to the current one.

2.4. Simulated Annealing

This method accepts any move which results in a solution of equal or greater quality than the previous move. If a move produces a poorer quality solution, the move is accepted probabilistically based on how much poorer the neighboring solution is and the current temperature (a parameter which decreases over time). The acceptable level of worsening solutions will decrease as the temperature decreases and the probability of moving to a worse solution will reduce over time [28].

There is a growing number of studies on selection hyper-heuristics combining a range of simple heuristic selection and move acceptance methods ([23], [29]). A recent theoretical study on selection hyper-heuristics showed that the mixing of simple move acceptance criteria could lead to an improved running-time complexity than using each move acceptance method standalone on some simple benchmark functions [27].

The next figure summarizes the classification of hyper-heuristics according to the different criteria mentioned earlier.
2.4 Hyper-heuristic Application:

Hyper-heuristics has been applied to different problems domain. The main characteristic of hyper-heuristic is to be able to operate across different problem types. Some applications are:

2.4.1 Traveling Salesman Problem (TSP)

The TSP is a special problem in computer science. The problem itself is conceptually very simple. The traveling salesman must visit every city in his territory exactly once and then return home to his starting point [40]. Given the cost of traveling between each pair of cities, the task is to arrange a complete tour that achieves minimum total cost. One possible search space for the TSP is a set of permutations of n cities. Any single permutation of n cities yields a solution, which is interpreted as a complete tour, with a return home being implied as soon as the last city in the list is visited. The size of this search space is n! [30]. The TSP has a long history. It was documented as early as 1759 by Euler, although not with the same name, who was interested in solving the knight's tour problem. A correct solution would have a knight visit each of the 64 squares. Due to the symmetry of the problem and the fact that a tour is a complete circuit, the space can
be reduced to \((n-1)!/2\) distinct tours. But any search algorithm that operates just on permutations must still treat \(n!\) possibilities. The Traveling Salesman Problem of a chess board exactly once in its tour. The term "traveling salesman" was first used in a German book from 1932 written by a veteran traveling salesman, The traveling salesman, how and what he should do to get commissions and be successful in his business [30].

2.4.2 Nurse scheduling problem (NSP)

NSP is the operations research problem of finding an optimal way to assign nurses to shifts, typically with a set of hard constraints which all valid solutions must follow, and a set of soft constraints which define the relative quality of valid solutions. The NSP determines how nurses are assigned to the shifts satisfying the hard constraints defined. Some examples of constraints are:

- A nurse does not work the day shift, night shift and late night shift on the same day (for obvious reasons).
- A nurse may go on a holiday and will not work shifts during this time.
- A nurse does not do a late night shift followed by a day shift the next day.

This problem is solved using many approaches. It uses both mathematically exact solutions and hyper-heuristic solution.

2.4.3 Timetabling problems

Timetabling problems are those kind of scheduling problems where the solution is a timetable follows the constraints used to define the problem. We can say that timetabling problems are a combinatorial optimization problem heavily inspired by the practice. Wren (1996) defined timetabling as “the process of allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives”. Based on this definition, we need to know whether there are sufficient resources available for the given event to take place at its specified time as well as which resources are allocated. The goal is to optimize some objective function depending on the application domain at hand [31].
There are many types of timetabling problems that differ in their structure, constraints and requirements such as educational timetabling, hospital timetabling and airport timetabling. Searching for better and optimal timetables as well as introducing additional requirements and constraints in the problem domain makes the researchers still trying to solve timetabling problem to produce the optimal solution. Timetabling problems were firstly solved using conventional methods where the work is done by people in a few days to several weeks. This solution is both time and labor consuming. Then number of software tools were developed to support this process. Still human factor is inserted in this method where final decisions must be taken by someone. For this, the need for automation arise.

Constraints in timetabling can be categorize into two category: hard and soft constraints. The hard constraints are those kind of constraints that must be satisfy and cannot be violated. While soft constraints are those kind where satisfying them is not essential but trying hard to satisfy them will produce a better solution. A general timetabling problem consists of assigning a number of events like courses, examinations, lectures, lab sessions etc. into a limited number of timeslots and rooms while minimizing the violations of a set of constraints. In general, the timetable problem has a set of events, E, a set of timeslots, T, and a set of hard constraints, C. The intention is to arrange all of the events, E, into the timeslots, T, in such a way that no hard constraint, C, is violated in order to produce a feasible timetable [31]. The basic terminology used in timetabling problems is summarized in Table 2.1.

**Table 2-1: Basic terminology used in timetabling [31]**

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>An activity to be scheduled. Examples include examinations and courses</td>
</tr>
<tr>
<td>Timeslot (period)</td>
<td>An interval of time in which events can be scheduled.</td>
</tr>
<tr>
<td>Resource</td>
<td>Resources required by events. Examples include rooms and equipment (i.e. projectors).</td>
</tr>
<tr>
<td>Constraint</td>
<td>A restriction to schedule the events. Examples include room capacity and specific timeslot.</td>
</tr>
<tr>
<td>Individual</td>
<td>A person who has to attend the events.</td>
</tr>
<tr>
<td>------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>Conflict</td>
<td>Two events are clashing with each other if they have at least a common individual and are scheduled in the same timeslot</td>
</tr>
</tbody>
</table>

2.5 Classification of Educational Timetabling Problems

Educational timetabling is classified into three classes: school timetabling, course timetabling and examination timetabling. They have the same idea of timetabling problems but still differ from each other in constraints, rules and requirements. Course and examination timetabling are sometimes called university timetabling. Normally in university timetabling, an initial solution is constructed using an appropriate heuristic, then the improvement is carried out using these meta-heuristics.

2.5.1 School Timetabling

This kind of timetabling is concerned with producing timetable containing all the materials of the school. The solution must assign materials to timeslots and teachers to a specific timeslot in a specific room. The solution also has to satisfy all the hard constraints and as much soft constraint as possible. The general constrains for this problems are for example: capacities, locations, teacher loads, rest time between two lessons and other personal preferences [31].

2.5.2 Course Timetabling

Carter and Laporte (1998) defined course timetabling as: “a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course sections or classes; events (individual meetings between students and teachers) are assigned to classrooms and times”. The idea behind course timetabling is to assign university courses to timeslot as well as assigning teachers and students to courses. Some of the hard constraints for course timetabling problem can be:

- Every student and every teacher must be assigned to only one room at a specific timeslot.
• Only one course is allowed to be assigned to a timeslot in each classroom.
• The size of the classroom should be enough for the number of students attending the course at a specific timeslot.

Some of the soft constraints for this problem can be:

• Students should more than one course on a day.
• Students should not have to attend more than two consecutive courses on a day.
• Students should not be scheduled to attend a course that is assigned to the last timeslot of the day [31].

Carter and Laporte (1998) decomposed the course scheduling problem into five sub problems i.e. course timetabling, class-teacher timetabling, student scheduling, teacher assignment and classroom assignment. Laporte and Desroches (1986) presented a problem of assigning students to course sections in a large engineering school.

2.5.3 Examination Timetabling

Carter and Laporte (1996) defined examinations timetabling as: “The assigning of examinations to a limited number of available time periods in such a way that there are no conflicts or clashes”. The underlying problem of examination timetabling is considered to be the same (in the basic definition of the problem) as the graph coloring problem and, hence it is an NP-hard problem (Burke, Kingston and deWerra, 2004; Qu et al., 2009, etc). The idea behind examination timetabling is to assign exams to a specific or limited number of timeslots and rooms with the aim of satisfying the constraints. Some of the hard constraints for examination timetabling problem can be:

• Certain examinations must be consecutive or must take place in a specific order (before/after each other).
• Examinations with the largest number of students should be scheduled earlier in the timetable to allow more time for marking.
• Examinations given by the same instructor, if scheduled in the same timeslots, have to be assigned to nearby classrooms.
• There should be no more than x conflicting examinations in y (>x) consecutive timeslots.

• Certain examinations must take place in a specific room.

• There must be enough seating capacity in the room for the number of students scheduled in it.

• No student should be required to sit two examinations simultaneously [31].

On the other hand some of the soft constraints are as follow:

• Students should not be scheduled to sit more than one examination in a day.

• Students should not be scheduled to sit examinations in two consecutive timeslots.

• Each student’s examinations should be spread as evenly as possible over the schedule.

• Some examinations may only be assigned within a particular set of timeslots.

• Examinations of the same length may be scheduled in the same room.

• Examinations must be scheduled to the rooms which are near to the relevant department.

• Examinations with questions in common must be scheduled in the same timeslot.

 Practically not all soft constraints can be satisfied yet minimizing the violation will increase the quality of the solution. In some situations, the examination timetabling problem becomes more difficult as these constraints conflict with one another, where satisfaction of one constraint can lead to a violation of another (Qu et al., 2009). For example, suppose we have a situation where we want to minimize the total examination period and at the same time we wish to spread out exams as much as possible [32]. In such a situation, satisfaction of one constraint will lead to poor quality solutions of the other constraint.

De Werra (1985) presented a formal approach for the examination timetabling problem based on a mathematical programming model. Let us consider the following notation [31]:

• $E_i$ is a collection of n examinations ($i = 1, \ldots, N$)
• T is the number of timeslots.
• \( C_{it} \) is the cost of scheduling examination i in timeslot T.
• \( Y_{it} = 1 \) if examination i is scheduled in timeslot T and 0 otherwise.
• \( X_{ij} = 1 \) if examination i clashes with examination j and 0 otherwise

A timetable is considered feasible if the following hard constraints are satisfied [31]:

- Every examination must be scheduled once
- No conflicting examinations should be scheduled in the same timeslot.

The problem is required to generate a timetable using T timeslots without violating the hard constraints. The objective is to minimize the cost of scheduling examination i in period T. This examination timetabling problem can be represented as (adopted from Terashima-Marín, 1998):

\[
\min \sum_{n=1}^{N} \sum_{t=1}^{T} C_{nt} Y_{nt} \tag{2.1}
\]

Subject to

\[
\sum_{t=1}^{T} Y_{nt} = 1 \quad n=1, 2, \ldots, N \tag{2.2}
\]

\[
\sum_{n=1}^{N} \sum_{m=1}^{N} \sum_{t=1}^{T} Y_{nt} Y_{mt} Y_{nm} = 0 \tag{2.3}
\]

The other hard constraint that can also be considered is concerned with room capacity. Let \( X_t \) be the maximum number (capacity) of examinations that can be scheduled in period t, so the hard constraints can be represented as [31]:

\[
\sum_{n=1}^{N} Y_{nt} \leq X_t \quad t=1, 2, \ldots, T \tag{2.4}
\]

The room capacity can be considered as a hard constraint or not consider. When the room capacity is a hard constraint this type of timetabling is called capacitated timetabling.
problems which is close to the real world problem description. The other type is called un-capacitated timetabling problem which is widely used in the literature.

The final solution of examination timetabling problems should guarantee that every student can take any examination that he required to. In order to solve those kind of problem some constraints should be satisfied and this degree of satisfaction establishes a quality measure for the timetable.

Those two kind of university timetabling, course timetabling and examination timetabling, are concerned with avoiding the constraint that one student assign to two courses or exams at the same timeslot. Both types are totally differ in the constraints they have to satisfy. For example, an exam timetable may allow multiple exams in one room unlike a course timetable. This because it is obviously not possible to have two different courses/lectures in the same room. With respect to user preferences, in course timetabling students are free to select their optional courses to suit their own course objectives. This is not the case with an exam timetable as the examinations contain registered students and, therefore we need to consider a clash free (hard constraint) timetable (among others) and student satisfaction (soft constraint) in producing the exam timetable [32]. Besides the differences in constraints, course and examination differ in the way in which they are constructed, which can be divided into process environment, modeling and scheduling instances [32]. In the process environment, normally, the course timetable is produced separately and independently by each school, unlike an exam timetable which is usually produced centrally by the academic office (McCollum, 2007; Burke et al., 1996). In modeling, for course timetabling, it is constructed based on the projected number of students that will taking the courses, while in exam timetabling it is generated based on the number of registered students on particular course (McCollum, 2007). In scheduling instances, exam and courses use different instances although it is from the same source (I.e. courses). Examination timetables are formed based on the offered courses. While, in course timetable we need to schedule the individual lectures, tutorial and labs from the offered course (McCollum, 2007). Although differences exist between the examination and course problem, the complexity of examination timetabling problem depends on the amount of freedom of choice on students selecting their course timetable (Laporte and
Desroches, 1984). The more freedom a student has increases the difficulty in producing a feasible examination timetable [32].

2.5.3.1 TDTP

Creating the table for graduation projects presentations is considered one type of examination timetabling. Graduation projects presentations considered as an important issue of the management of a universities. The number of graduating students may be very large, causing that graduation procedure may split into several sessions, with separate committees and possibly running in different days. The idea behind this type of timetabling is to come up with timetable that assign each group of students, sharing the same project with a session, room and a committee satisfying the constraints and objectives. Usually there are two sessions per day (morning and afternoon) and the duration depends on the number of students assigned [33]. Students must be assigned exactly to one session, whereas faculty members can participate to more than one committee (or none) [33]. The number of members of the committee and the number of the students having the same project is determined by university. There are many hard and soft constraints that may be determined by each university differently.

2.6 The International Timetabling Competition (ITC):

It’s a competition organized by Meta-heuristics Network which aims to encourage the research and practice in timetabling problems. The problems proposed at the competition are to develop solutions based on real-world data to generate timetables in consideration of different constraints. [33]. ITC was held three times—at the time of writing—with different orientation each time, in the years 2002, 2007 and 2011. In 2002 the competition was designed in order to promote research into automated methods for timetabling. [34]. Building on its success the second competition was conducted to further develop interest in the general area of educational timetabling while providing researchers with models of the problems faced which incorporate an increased number of real world constraints [35]. The third ITC, in 2011, was devoted to High School Timetabling. [33]. The participants are always provided with real-world datasets and they compete in three rounds, where each round has different problem specifications. The
overall aim of the competition was to create better understanding between researchers and practitioners by allowing emerging techniques to be trailed and tested on real world models of timetabling problems [35]. The success of this competition is a conclusive proof of the importance and difficulty of the timetabling problem.
CHAPTER 3

PROBLEM STATEMENT

3.1 Introduction:

This chapter presents a detailed explanation of our problem, TDTP. It also shows how difficult is the process of creating timetables using classical methods. It also contains a list of the hard and the soft constraints of the problem.

3.2 Problem Description

The thesis examination panel timetabling (TEPT) problem is concerned with assigning each group of graduating students to different sessions and compositing one committee for each session. The number of students in each group, number of sessions and the number of examiners is defined by the department. Each group of students must be examined only once, while the examiners can examine more than one group. The solution is considered feasible if all hard constraints are respected.

The main entities of this problem are:

- **Projects**
  
  It is a list containing a predefined distinct projects taken by each group of students.

- **Examiners**
  
  It is a list containing the names of the faculty members who are qualified enough to be part of the committee.

- **Rooms**
  
  It is a list of chosen rooms where the examination panel can take place.

- **Sessions**
  
  It is a list of scheduled timeslots that the project can be assigned to.
3.3 Solving timetabling problem using classical methods

The TEPT is a computationally exhausting task. Using the fastest supercomputer –at the time of writing- named Sequoia which runs 17.2 petaflops will give the following results:

1. Finding all possible combinations of timetables resulting by assigning three examiners to each project. The equation is as follow:

\[
\text{Number of possible combinations} = (2^m)^n \quad (3.1)
\]

Where \( m = \) Total number of examiners

\( n = \) Total number of projects

<table>
<thead>
<tr>
<th>Total number of projects</th>
<th>Total number of examiners</th>
<th>Number of possible combinations</th>
<th>Time required by Sequoia</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10</td>
<td>(2.04 \times 10^{90})</td>
<td>(3.8 \times 10^{66}) years</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>(1.07 \times 10^{301})</td>
<td>(1.76 \times 10^{278}) years</td>
</tr>
<tr>
<td>60</td>
<td>25</td>
<td>(3.5 \times 10^{451})</td>
<td>(6.5 \times 10^{427}) years</td>
</tr>
<tr>
<td>75</td>
<td>30</td>
<td>(2.1 \times 10^{677})</td>
<td>(3.8 \times 10^{635}) years</td>
</tr>
</tbody>
</table>

2. Finding all possible combinations of timetables resulting by assigning projects to sessions. The equation is as follow:

\[
\text{Number of possible combinations} = (2^m)^n \quad (3.2)
\]

Where \( m = \) Total number of sessions

\( n = \) Total number of projects
Table 3-2: Time required to solve timetabling problem using Sequoia

<table>
<thead>
<tr>
<th>Total number of projects</th>
<th>Total number of sessions</th>
<th>Number of possible combinations</th>
<th>Time required by Sequoia</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>6</td>
<td>$1.5 \times 10^{54}$</td>
<td>$2.8 \times 10^{30}$ years</td>
</tr>
<tr>
<td>50</td>
<td>6</td>
<td>$2.03 \times 10^{90}$</td>
<td>$3.7 \times 10^{66}$ years</td>
</tr>
<tr>
<td>60</td>
<td>9</td>
<td>$3.6 \times 10^{162}$</td>
<td>$6.6 \times 10^{138}$ years</td>
</tr>
<tr>
<td>75</td>
<td>12</td>
<td>$8.5 \times 10^{270}$</td>
<td>$1.6 \times 10^{247}$ years</td>
</tr>
</tbody>
</table>

It is clear from tables (3-1, 3-2) that solving such problem using classical method is not feasible, since only finding all possible combinations of timetables takes millions of years. In addition to that, one needs to select the optimal timetables manually, which is nearly impossible. There rise the need for automation (Heuristics).

3.4 Constraints of the problem

To find the optimal solution using heuristic methods various constraints must be satisfied. The constraints are divided into: hard constraints and soft constraints, which are defined according to the current situation. The hard constraints are:

1. Examiners constraints:
   - Number of assigned examiners to a panel must not exceed a defined number by the department.
   - The examiner must not be assigned to two projects at the same session.
   - The examiner must not be assigned to a project they refuse be in.
   - The examiner must not be assigned to a session they are not available at.

2. Sessions and rooms constraints:
   - All projects must be assigned to a session.
   - Only one session must be assigned to a project.
   - Maximum number of projects assigned to one session must be as defined by the department.
• Unavailable rooms must not be assigned.

These constraints must always be satisfied, their violation is not accepted.

On the hand, the soft constraints are:

1. **Examiners constraints:**
   • Assigning an examiner to a project panel while they prefer not to be in.
   • Assigning an examiner into all session.
   • Similar projects are preferred to be placed at the same panel with the same committee.
   • Projects which have the same supervisor are preferred to be placed at the same panel.
   • Examiners should not be scheduled to contribute in two consecutive sessions.

2. **Sessions and rooms constraints:**
   • There must be enough seating capacity for the committee members.
   • Some panels may only be assigned within a particular set of sessions.
   • Satisfaction of these constraints is desirable but not obligatory.

These constrains are not necessary in producing a feasible solution. It just increase the quality of the solution obtained.
CHAPTER 4

METHODOLOGY

4.1 Overview

This chapter gives a deep insight and describes the methods and concepts in details that used to implement the solution of the TEPT problem. It also demonstrates the steps used to compose an optimal timetable.

4.2 Introduction

The methodology of solving the TDTP was implemented by using Microsoft excel 2010. The code was written in Visual Basic for Application (VBA). This solution ran on Windows 7 machine with 4 Intel core i7 physical cores (1.8 GHz) with 6 GB RAM.

A user manual is appended at appendix D.

4.3 Problem Instances

The input data has been collected from a real world problem. List of projects, examiners, rooms and sessions of the year 2015-2016 was obtained from U of K. Summary of the problem instances is shown in table 4-1.

<table>
<thead>
<tr>
<th>Table 4-1: Data Instances summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of projects</td>
</tr>
<tr>
<td>Number of supervisors</td>
</tr>
<tr>
<td>Number of examiners</td>
</tr>
<tr>
<td>Number of rooms</td>
</tr>
<tr>
<td>Number of sessions</td>
</tr>
<tr>
<td>Number of groups per session</td>
</tr>
<tr>
<td>Number of students</td>
</tr>
</tbody>
</table>
The lists were then organized as excel sheets. Microsoft excel was used because of the following reasons:

- Excel is a ready-to-use and familiar environment. There is no need to install new packages nor application programming interfaces.
- Excel comes with an integrated programming language. Excel can compile and execute visual basic for applications (VBA) code after editing it in the visual basic editor which is a convenient environment for developing code.
- Excel enable the users to have all input data, the code, all Calculation results and visualization in one place.
- Excel is a widely spread application.

The input sheets are organized as shown:

- **Sheet 1: Supervisors _Examiners**
  This sheet contains all the names of faculty members who are qualified enough to be member of the committee. It also contains the supervisors of the projects.

- **Sheet 2: Projects**
  This sheet contains all the projects listed by the department. Each project with its supervisor and group of students.

- **Sheet 3: Rooms**
  This sheet contains all the rooms where exam sessions will take place. It also shows the availability of each room for each session.

- **Sheet 4: Suitability**
  This sheet shows to which extent an examiner is interested in each project. For sake of research, this sheet was filled randomly.

- **Sheet 5: Availability**
  This sheets shows whether an examiner is available at each defined session or not. For sake of research, this sheet was filled randomly.

These sheets are appended in appendix B.
4.4 Data setup

The validity of the collected data was checked as shown by the following steps:

- **Step 1:**
  Checking the existence of the five sheets mentioned in 4.2. If any of them is nonexistent then an error message appears on the screen.

- **Step 2:**
  Checking the validity of data contents of sheet supervisors _examiners by looking for names redundancies. Then storing these names in an array.

- **Step 3:**
  Storing the names of the projects from sheet projects in an array.

- **Step 4:**
  Checking the validity of data contents of sheet projects by comparing the names of supervisors in this sheet by the names in sheet supervisors _examiners.

- **Step 5:**
  Storing the name of each project with its supervisor and students in an array.

- **Step 6:**
  Storing the sessions from sheet rooms in an array. Then, names of the rooms and availability of these rooms in each session were stored in separate arrays.

- **Step 7:**
  Checking the validity of the data contents of sheet suitability by comparing its inputs to sheets projects and supervisors _examiners. Then suitability of each project to examiners were stored in separate array.
Step 8:

Checking the validity of the data contents of sheet availability by comparing its inputs to sheets rooms and supervisors _examiners. Then availability of each examiner in each session were stored in separate array.

4.5 Solution Representation:

Two arrays were used for the solution representation one for sessions and one for examiners.

4.5.1 Sessions solution representation

It is three dimensional array used to assign projects to sessions and rooms. Its size is specified by number of projects, number of rooms and number of sessions. The elements of the array are either 0 or 1. 0 means that the project is not assigned to the session, while 1 means that it is assigned as shown in table 4-2.

<table>
<thead>
<tr>
<th>Room 2</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 2</td>
<td>Session 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Project 1

<table>
<thead>
<tr>
<th>Room 2</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 2</td>
<td>Session 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Project 2

<table>
<thead>
<tr>
<th>Room 2</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 2</td>
<td>Session 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Project 3

4.5.2 Examiners solution representation

It is a two dimensional array used to assign examiners to projects. Its size is specified by number of projects and number of examiners in the committee which is three as specified by the department. The elements of the array are in the range [1, number of examiners]. Where each number in this interval is the index of the supervisor in supervisors _examiners sheet as shown in table 4-3.

<table>
<thead>
<tr>
<th>Room 2</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 2</td>
<td>Session 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Project 1

<table>
<thead>
<tr>
<th>Room 2</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 2</td>
<td>Session 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Project 2

<table>
<thead>
<tr>
<th>Room 2</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session 2</td>
<td>Session 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Project 3
Even though these two solution representations seem to be separated, some constraints may affect both of them at the same time. Sometimes they need to satisfy the same constraint. Making changes on one of the solution representations may require some changes on the other.

4.6 Solution steps

![Flow chart for solution steps]

**Figure 4-1:** Flow chart for solution steps
4.7 Initialization:

Two methods of initialization were implemented.

4.7.1 Constant value initialization

All the elements of the array were set to constant value. For the session solution representation of our problem this value was 0, which means that no project was assigned to any session nor room as shown in the next pseudo code:

```
Algorithm 1: Initializing with constant value
For i = 1 to number of projects Do
    For j = 1 to number of rooms Do
        For k = 1 to number of sessions Do
            Array (i, j, k) = 0
        End
    End
End
```

The resulting timetable is shown in table 4-4.

**Table 4-4:** Example of session solution representation after initialization

<table>
<thead>
<tr>
<th></th>
<th>Room 2</th>
<th>Room 1</th>
<th>Room 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Session 2</td>
<td>Session 1</td>
<td>Session 2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

On the other hand, for the examiner solution representation of our problem this value was 1, which means that all the projects were assign to the same examiner who has the index
1 in supervisors_examiners sheet, and no other examiner in the committee as shown in the next pseudo code:

```
Algorithm 2: Initializing with constant value
For i = 1 to number of projects Do
    For j = 1 to number of examiners in the committee Do
        Array (i, j) = 1
    End
End
```

The resulting timetable is shown in table 4-5.

```
<table>
<thead>
<tr>
<th>Examiner 3</th>
<th>Examiner 2</th>
<th>Examiner 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Project 3</td>
</tr>
</tbody>
</table>
```

**Table 4-5:** Example of examiners solution representation

### 4.7.2 Random initialization

The elements of the array were set to random values. The random values range differ according to the solution representation. For sessions solution representation of our problem the random values were chosen randomly whether 0 or 1 as shown in the next pseudo code:

```
Algorithm 3: Initializing with random values
For i = 1 to number of projects Do
    For j = 1 to number of rooms Do
        For k = 1 to number of sessions Do
            Array (i, j, k) = Random (0, 1)
        End
    End
End
```
On the hand, for the examiner solution representation of our problem the random values were chosen randomly from the following set:

\[ \text{Random\_value} = \{1, 2, 3, 4, \ldots, 47\} \]

Where 47 is the number of supervisors_examiners. As shown in the next pseudo code:

<table>
<thead>
<tr>
<th>Algorithm 4: Initializing with random values</th>
</tr>
</thead>
<tbody>
<tr>
<td>For ( i = 1 ) to number of projects Do</td>
</tr>
<tr>
<td>\hspace{1cm} For ( j = 1 ) to number of examiners in the committee Do</td>
</tr>
<tr>
<td>\hspace{2cm} Array ( (i, j) ) = Random ( (1, 2, 3, \ldots, \text{number of examiners}) )</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>End</td>
</tr>
</tbody>
</table>

4.8 Low level heuristics

These are the methods that are used to manipulate the solution representation to reduce the number of violations of constraints. Six methods of low level heuristics were used for this problem.

- **LLH1**: This heuristic selects random project then checks whether it is assigned to a certain session or not. If it is assigned it un-assigns it and vice versa as shown in figure 4-2.
Figure 4-2: Example of changing assignment randomly

- **LLH2**: This heuristic runs LLH1 twice which will result in more variance by changing the assignment of more projects.
- **LLH3**: This heuristic selects random project then checks whether it is assigned to a certain session or not. If it is assigned it un-assigns as shown in figure 4-3.

Figure 4-3: Example of un-assigning randomly

- **LLH4**: This heuristic selects random projects and changes one of its assigned examiners to another random examiner as shown in figure 4-4.
Figure 4-4: Example of changing assignment randomly

- **LLH5**: This heuristic runs LLH3 twice which will result in more variance by changing the assignment of more examiners.
- **LLH6**: This heuristic randomly swaps two examiners as shown in figure 4-5.

Figure 4-5: Example of randomly swap

### 4.8.1 Utilization of low level heuristics

The effectiveness of each low level heuristic at improving the solution is called utilization. It is measured as shown in the following pseudo code:

#### Algorithm 5: Utilization

Let LLH be set of low level heuristic = \{LLH1, LLH2,…,LLHn\}

Where n is the total number of low level heuristics

Let Util (1 to n) be array of utilization
4.9 Evaluation method

In the evaluation process it is desired to measure two aspects of the performance: the cost and the quality of the solution.

- The cost is calculated using the following equation [37]:

\[
\text{total cost} = \sum_{i=1}^{k} w_i n_i(S)
\]

(4.1)

Where:

\( W_i = \) Cost weight (penalty value), hard constraints heavily penalized

\( n_i(S) = \) Number of violation

\( S = \) Solution representation

- The quality of the solution is calculated using the following equation [37]:

\[
\text{Qual} = \frac{1}{1 + \text{total cost}}
\]

(4.2)

If Qual = 1 this means that the solution is fully satisfying all the constraints.

In this project the cost are not weighted because we are only concerned with generating feasible solutions.
4.10 Hyper-heuristic

There are two types of hyper-heuristics: Selecting heuristics and generating new heuristics. In this project we are concerned with selecting heuristic, which consist of two stages as shown in figure 4.6.

![Hyper-heuristic Diagram](image)

**Figure 4-6:** Selection hyper-heuristic [38]

### 4.10.1 Heuristic selection

It is the process of selecting low level heuristic from a set of predefined low level heuristics which will generate new solution. In this project simple random was used as shown in the following pseudo code:

```plaintext
Algorithm 6: Select heuristics

Let LLH be set of low level heuristic = {LLH1, LLH2,…,LLHn}

Where n is the total number of low level heuristics

h = Select Randomly (LLH)
```
4.10.2 Move acceptance

It is the process that decides whether to accept the new solution or not. In this project, we used three methods of move acceptance:

4.10.2.1 Accept all moves

This method accepts all generated solutions even if they are worse than the current solution as shown in the next pseudo code:

<table>
<thead>
<tr>
<th>Algorithm 7: Accept all moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $S_{\text{old}}$ be previous solution</td>
</tr>
<tr>
<td>Let $S_{\text{cur}}$ be current solution</td>
</tr>
<tr>
<td>$S_{\text{old}} = S_{\text{cur}}$</td>
</tr>
</tbody>
</table>

4.10.2.2 Accept only improving moves

This method accepts only the solutions that are better than the current solution as shown in the next pseudo code:

<table>
<thead>
<tr>
<th>Algorithm 8: Accept only improving moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $S_{\text{old}}$ be previous solution</td>
</tr>
<tr>
<td>Let $S_{\text{cur}}$ be current solution</td>
</tr>
<tr>
<td>Let $f(s)$ be total cost of the solution as in equation 4.1</td>
</tr>
<tr>
<td>If $f(S_{\text{cur}}) &lt; f(S_{\text{prev}})$ then</td>
</tr>
<tr>
<td>$S_{\text{prev}} = S_{\text{cur}}$ // accept</td>
</tr>
<tr>
<td>Else</td>
</tr>
</tbody>
</table>
Since this method only accept improved solution, at the end it will find the best solution.

4.10.2.3 Accept Improving or equal moves

This method accepts only the solutions that are better than or the same as the current solution as shown in the next pseudo code:

<table>
<thead>
<tr>
<th>Algorithm 9: Accept improving or equal moves</th>
</tr>
</thead>
<tbody>
<tr>
<td>Let $S_{\text{old}}$ be previous solution</td>
</tr>
<tr>
<td>Let $S_{\text{cur}}$ be current solution</td>
</tr>
<tr>
<td>Let $f(s)$ be total cost of the solution as in equation 4.1</td>
</tr>
<tr>
<td>If $f(S_{\text{cur}}) \leq f(S_{\text{prev}})$ then</td>
</tr>
<tr>
<td>$S_{\text{prev}} = S_{\text{cur}}$  // accept</td>
</tr>
<tr>
<td>Else</td>
</tr>
<tr>
<td>$S_{\text{prev}} = S_{\text{prev}}$  // reject</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

Mainly, there are two types of stopping criteria that control the move acceptance process:

- **Time**

  In this type, the process of move acceptance will be repeated for a specific period of time.

- **Number of iterations**

  In this types, the process of move acceptance will be repeated for a specific number of iterations.

  In this project number of iterations was used. Changing number of iterations result in different values of cost.
CHAPTER 5

RESULTS

5.1 Overview

This chapter presents the results and give explanations to them. It also evaluates these results against the aims and objectives outlined in chapter 1.

5.2 Final results

The final results strongly depend on the number of violations that appear in the solution (the cost). Minimizing the violations present better solution.

There are three factors that affect the cost of the final solution:

- The initialization process
- The type of low level heuristic
- The method used for move acceptance process

Results using random initialization with different combinations of heuristic with 10000 iterations is shown in table 5-1:

<table>
<thead>
<tr>
<th></th>
<th>Accept all</th>
<th>Accept only improved</th>
<th>Accept improved or equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLH1 + LLH4</td>
<td>2043</td>
<td>1809</td>
<td>1825</td>
</tr>
<tr>
<td>LLH3 + LLH4</td>
<td>171</td>
<td>67</td>
<td>66</td>
</tr>
<tr>
<td>LLH1 + LLH3+LLH5</td>
<td>222</td>
<td>242</td>
<td>260</td>
</tr>
<tr>
<td>LLH1 + LLH5+LLH6</td>
<td>2046</td>
<td>2000</td>
<td>1805</td>
</tr>
</tbody>
</table>
Results using constant initialization with different combinations of heuristic with 10000 iterations is shown in table 5-2:

Table 5-2: Results obtained from using constant initialization with different heuristic

<table>
<thead>
<tr>
<th>Combination</th>
<th>Accept all</th>
<th>Accept only improved</th>
<th>Accept improved or equal</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLH1 + LLH4</td>
<td>2057</td>
<td>1867</td>
<td>1891</td>
</tr>
<tr>
<td>LLH3 + LLH4</td>
<td>172</td>
<td>70</td>
<td>67</td>
</tr>
<tr>
<td>LLH1 + LLH3 + LLH5</td>
<td>205</td>
<td>174</td>
<td>168</td>
</tr>
<tr>
<td>LLH1 + LLH5 + LLH6</td>
<td>273</td>
<td>467</td>
<td>479</td>
</tr>
<tr>
<td>LLH3 + LLH4 + LLH5 + LLH6</td>
<td>178</td>
<td>77</td>
<td>70</td>
</tr>
<tr>
<td>LLH1 + LLH2 + LLH3 + LLH5 + LLH6</td>
<td>287</td>
<td>472</td>
<td>477</td>
</tr>
</tbody>
</table>

From table 5-1 and table 5-2 the following things were noticed:

- LLH1 seems not performing well for our problem because it assigns too many sessions to projects where only one session is required. It may also assigns too many projects in one session while only six is needed.
- Accept all moves always result in greater cost than accept only improved moves and accept improved or equal moves.
- Initialize the solution representation with random values is better than initialize with constant value.
Starting with a random solution representations then using LLH3 for the first solution representation and LLH4 for the second solution representation with using improved or equal moves for move acceptance, results in the best solution. This best solution is then with 10000 iterations used to improve the cost. Some sample data from the output is shown in table 5-3.

<table>
<thead>
<tr>
<th>Iteration number</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2190</td>
</tr>
<tr>
<td>34</td>
<td>994</td>
</tr>
<tr>
<td>100</td>
<td>322</td>
</tr>
<tr>
<td>1000</td>
<td>152</td>
</tr>
<tr>
<td>2033</td>
<td>100</td>
</tr>
<tr>
<td>4000</td>
<td>73</td>
</tr>
<tr>
<td>8015</td>
<td>66</td>
</tr>
<tr>
<td>10000</td>
<td>65</td>
</tr>
</tbody>
</table>

The output of the 10000 iterations is shown in figure 5-1

From figure 5-1 it is clear that the cost decreases very fast at the first iterations. Then it decreases slowly until it reach a point where no noticeable changes occur. This means that at this point the heuristic cannot make further improvement to the cost.
5.3 Comparison between low level heuristics:

A comparison was done between LLHs for each solution representations using utilization. The utilization means how much a specific heuristic make an improvement to the solution representation.

- Utilization for sessions solution representation

![Pie chart showing utilization for LLH1, LLH2, and LLH3 for sessions solution representation]

**Figure 5-2:** shows the utilization for LLH1, LLH2 and LLH3 for sessions solution representation

- Utilization For examiners solution representation
Figure 5.3: shows the utilization for LLH1, LLH2 and LLH3 for examiners solution representation

Sample from the output timetable is shown in appendix C. From that it is concluded that some violations appears, for example:

- Some projects were not assigned to sessions.
- Most of the projects were assigned to one rooms.
- Some examiners were repeated in one committee.

Final words this solution is not the optimal solution.
CHAPTER 6

CONCLUSION

6.1 Overview

This chapter gives a conclusion of the work done for this project. It also contains what can be done in the future to improve the solution obtained.

6.2 Summary of work

The overall aim of the project presented in this thesis was to generate the thesis defense timetable and trying to find a combination of low level heuristics and move acceptance methods that will result in better solution. The data was collected from real world. It was taken from U of K, and a platform was created for this problem.

It is concluded from this research that combing the heuristic that selects random project then if it is assigned it un-assigns it and accept improved or equal solutions move acceptance, results in the best solution from all. This heuristic with this move acceptance is not sufficient because it does not give zero cost yet, it results in a good solution because it reduce the cost from its start to its end.

This project focuses on producing a feasible solution that is why soft constraint were not taken into consideration.

6.3 Future work

- Trying to find out new selection methods rather than simple random.
- Trying to find out new move acceptance methods or LLHs that result in no violation.
- Trying to find out more efficient solution representation.
- Adding the soft constraint to the problem to improve the equality of the solution.
References


APPENDIX A: THE CODE

' University of Khartoum
' Faculty of Engineering
' Department of Electrical and Electronic Engineering
' Thesis Defense Timetabling Problem using hyper-heuristics
' By: Elham yasir and Maha Ali
' Supervisor: Dr. Ahmed Kheiri

Sub main_Panel()
    Dim FSO
    Dim inputFile As String
    Dim inputFilePath As String
    Dim outputFilePath As String
    Dim inputFile As Workbook
    Dim newFile As Workbook
    inputFile = InputBox("Enter Input Filename include extension. e.g. Panel_2015-2016.xls")
    inputFilePath = ActiveWorkbook.Path & "\input\" & inputFileName
    outputFilePath = ActiveWorkbook.Path & "\output\Output_" & inputFileName
    Set FSO = CreateObject("Scripting.FileSystemObject")
    If Not FSO.FileExists(inputFilePath) Then
        MsgBox "ERROR!! Specified File Not Found", vbInformation, "Not Found"
    Else
        'Create Object
        Set FSO = CreateObject("Scripting.FileSystemObject")
        'Copying If the Same File is Not Located in the Destination Folder
        If Not FSO.FileExists(outputFilePath) Then
            FSO.CopyFile inputFilePath, outputFilePath, True
            Set newFile = Workbooks.Open(outputFilePath)
            Set inputFile = Workbooks.Open(inputFilePath)
            If CreatePanel(newFile, inputFile) = False Then
                newFile.Save
APPENDIX A: THE CODE

newFile.Close
inputFile.Close
Kill outputFilePath
Else
newFile.Save
newFile.Close
inputFile.Close
End If

Else
MsgBox "ERROR2!! Specified File Already Exists In The Destination Folder", vbExclamation, "File Already Exists"
End If
End If
End Sub

Function CreatePanel(file_o As Workbook, file_i As Workbook) As Boolean

Dim number_of_supervisors_examiners As Integer
Dim number_of_projects As Integer
Dim i As Integer, j As Integer, k As Integer
Dim is_valid As Boolean
Dim number_of_periods As Integer
Dim number_of_rooms As Integer

'check that all sheets exist
'check that sheet projects exist

If Not SheetExists("projects", file_i) Then
    MsgBox "ERROR!! cannot find projects sheet"
    CreatePanel = False
    Exit Function
End If

'check that sheet examiners_supervisors exist
APPENDIX A: THE CODE

If Not SheetExists("examiners_supervisors", file_i) Then
    MsgBox "ERROR!! cannot find examiners_supervisors sheet"
    CreatePanel = False
    Exit Function
End If

'check that sheet rooms exist

If Not SheetExists("rooms", file_i) Then
    MsgBox "ERROR!! cannot find rooms sheet"
    CreatePanel = False
    Exit Function
End If

'check that sheet suitability exist

If Not SheetExists("suitability", file_i) Then
    MsgBox "ERROR!! cannot find suitability sheet"
    CreatePanel = False
    Exit Function
End If

'check that sheet availability exist

If Not SheetExists("availability", file_i) Then
    MsgBox "ERROR!! cannot find availability sheet"
    CreatePanel = False
    Exit Function
End If

'input supervisors & examiners

number_of_supervisors_examiners = 0

Do While
    file_i.Sheets("examiners_supervisors").Cells(number_of_supervisors_examiners + 2, 1).Value <> ""
A-3
APPENDIX A: THE CODE

number_of_supervisors_examiners =
number_of_supervisors_examiners + 1
Loop

ReDim names_of_supervisors_examiners(1 To
number_of_supervisors_examiners) As String

For i = 1 To number_of_supervisors_examiners
    For j = 1 To i - 1
        If names_of_supervisors_examiners(j) =
        file_i.Sheets("examiners_supervisors").Cells(i + 1, 1).Value Then
            MsgBox "ERROR!! " &
            file_i.Sheets("examiners_supervisors").Cells(i + 1, 1).Value & " has
            already been added to the list"
            CreatePanel = False
            Exit Function
        End If
    Next j
    names_of_supervisors_examiners(i) =
    file_i.Sheets("examiners_supervisors").Cells(i + 1, 1).Value
Next i

'input projects
number_of_projects = 0
Do While file_i.Sheets("projects").Cells(number_of_projects + 2, 1).Value <> ""
    number_of_projects = number_of_projects + 1
Loop

'check "projects" sheet is valid
For i = 1 To number_of_projects
    is_valid = False
    For j = 1 To number_of_supervisors_examiners
        If names_of_supervisors_examiners(j) =
        file_i.Sheets("projects").Cells(i + 1, 3).Value Then
            is_valid = True
            Exit For
        End If
    Next j
    If is_valid = False Then
MsgBox "ERROR!! cannot find " & file_i.Sheets("projects").Cells(i + 1, 3).Value & " in projects sheet"
CreatePanel = False
Exit Function
End If
Next i

'input the data of "projects" sheet
ReDim data_of_projects_sheet(1 To number_of_projects, 1 To 3) As String
For i = 1 To number_of_projects
    For j = 1 To 3
data_of_projects_sheet(i, j) = file_i.Sheets("projects").Cells(i + 1, j).Value
    Next j
Next i

'input rooms
number_of_periods = 0
Do While file_i.Sheets("rooms").Cells(2, number_of_periods + 2).Value <> ""
    number_of_periods = number_of_periods + 1
Loop

ReDim date_and_time_of_periods(1 To 3, 1 To number_of_periods) As String
For i = 1 To 3
    For j = 1 To number_of_periods
date_and_time_of_periods(i, j) = file_i.Sheets("rooms").Cells(i + 1, j + 1).Value
    Next j
Next i

number_of_rooms = 0
Do While file_i.Sheets("rooms").Cells(number_of_rooms + 5, 1).Value <> ""
    number_of_rooms = number_of_rooms + 1
Loop
APPENDIX A: THE CODE

ReDim names_of_rooms(1 To number_of_rooms) As String
For i = 1 To number_of_rooms
    For j = 1 To i - 1
        If names_of_rooms(j) = file_i.Sheets("rooms").Cells(i + 4, 1).Value Then
            MsgBox "ERROR!! " & file_i.Sheets("rooms").Cells(i + 4, 1).Value & " has already been added to the list"
            CreatePanel = False
            Exit Function
        End If
    Next j
    names_of_rooms(i) = file_i.Sheets("rooms").Cells(i + 4, 1).Value
Next i

ReDim availability_of_rooms(1 To number_of_rooms, 1 To number_of_periods) As Integer
For i = 1 To number_of_rooms
    For j = 1 To number_of_periods
        availability_of_rooms(i, j) = file_i.Sheets("rooms").Cells(i + 4, j + 1).Value
    Next j
Next i

'ReDim exams_per_period(1 To number_of_periods) As Integer
For i = 1 To number_of_periods
    exams_per_period(i) = file_i.Sheets("rooms").Cells(4, i + 1).Value
Next i

'check "suitability" sheet is valid
For i = 1 To number_of_projects
    is_valid = False
    For j = 1 To number_of_projects
        If data_of_projects_sheet(j, 1) = file_i.Sheets("suitability").Cells(i + 2, 1) And _
APPENDIX A: THE CODE

data_of_projects_sheet(j, 2) =
file_i.Sheets("suitability").Cells(i + 2, 2) And _
data_of_projects_sheet(j, 3) =
file_i.Sheets("suitability").Cells(i + 2, 3) Then
    is_valid = True
    Exit For
End If
Next j
If is_valid = False Then
    MsgBox "ERROR!! Some data of the " &
    file_i.Sheets("suitability").Cells(i + 2, 2).Value & " project is not correct"
    CreatePanel = False
    Exit Function
End If
Next i

'check name_of_supervisors_examiners
For i = 1 To number_of_supervisors_examiners
    is_valid = False
    For j = 1 To number_of_supervisors_examiners
        If names_of_supervisors_examiners(j) =
        file_i.Sheets("suitability").Cells(2, i + 3).Value Then
            is_valid = True
            Exit For
        End If
    Next j
    If is_valid = False Then
        MsgBox "ERROR!! cannot find " &
        file_i.Sheets("suitability").Cells(2, i + 3).Value & " in suitability sheet"
        CreatePanel = False
        Exit Function
    End If
Next i
Next i

ReDim suitability_of_projects_for_examiners(1 To number_of_projects, 1 To number_of_supervisors_examiners) As Integer
For i = 1 To number_of_projects
APPENDIX A: THE CODE

For j = 1 To number_of_supervisors_examiners
    suitability_of_projects_for_examiners(i, j) =
        file_i.Sheets("suitability").Cells(i + 2, j + 3).Value
Next j
Next i

'check "availability" sheet is valid

'check the date and time
For i = 1 To number_of_periods
    is_valid = False
    For j = 1 To number_of_periods
        If date_and_time_of_periods(1, j) =
            file_i.Sheets("availability").Cells(2, i + 1) And _
            date_and_time_of_periods(2, j) =
            file_i.Sheets("availability").Cells(3, i + 1) Then
            is_valid = True
            Exit For
        End If
    Next j
    If is_valid = False Then
        MsgBox "ERROR!! Some data of the time periods of the availability sheet is not correct "
        CreatePanel = False
        Exit Function
    End If
Next i

'check name_of_supervisors_examiners
For i = 1 To number_of_supervisors_examiners
    is_valid = False
    For j = 1 To number_of_supervisors_examiners
        If names_of_supervisors_examiners(j) =
            file_i.Sheets("availability").Cells(i + 3, 1).Value Then
            is_valid = True
            Exit For
        End If
    Next j
Next i
End If
Next j
If is_valid = False Then
    MsgBox "ERROR!! cannot find " & file_i.Sheets("availability").Cells(i + 3, 1).Value & " in availability sheet"
    CreatePanel = False
    Exit Function
End If
Next i

ReDim availability_of_examiners(1 To number_of_supervisors_examiners, 1 To number_of_periods) As Integer
    For i = 1 To number_of_supervisors_examiners
        For j = 1 To number_of_periods
            availability_of_examiners(i, j) = file_i.Sheets("availability").Cells(i + 3, j + 1).Value
        Next j
    Next i

''''''''''''''''''''''''''''''''''' OPTIMISATION
'''''''''''''''''''''''''''''''''''

ReDim solution(1 To number_of_projects, 1 To number_of_rooms, 1 To number_of_periods) As Integer
ReDim solution_exam(1 To number_of_projects, 1 To 3) As Integer

ReDim solution_prev(1 To number_of_projects, 1 To number_of_rooms, 1 To number_of_periods) As Integer
ReDim solution_exam_prev(1 To number_of_projects, 1 To 3) As Integer

ReDim solution_best(1 To number_of_projects, 1 To number_of_rooms, 1 To number_of_periods) As Integer
ReDim solution_exam_best(1 To number_of_projects, 1 To 3) As Integer

Dim cost_new As Long, cost As Long, cost_best As Long, iter As Long
Dim best_iter As Long
Dim h As Integer

init solution, number_of_projects, number_of_rooms, number_of_periods

init2 solution_exam, number_of_projects, number_of_supervisors_examiners

'******************************************************************
'evaluation
    cost = Evaluate(solution(), solution_exam(), number_of_projects, number_of_rooms, number_of_periods, exams_per_period(), availability_of_rooms(), suitability_of_projects_for_examiners(), availability_of_examiners())
'******************************************************************

copyFromTo solution, solution_prev, number_of_projects, number_of_rooms, number_of_periods

copyFromTo_exam solution_exam, solution_exam_prev, number_of_projects

cost_best = cost
best_iter = 0

For iter = 1 To 10000
    Randomize
    h = Int(4 * Rnd) + 1
    If h = 1 Then
        heuristic1 solution, number_of_projects, number_of_rooms, number_of_periods
    ElseIf h = 2 Then
        heuristic2 solution, number_of_projects, number_of_rooms, number_of_periods
    ElseIf h = 3 Or h = 4 Then
        heuristic3 solution, number_of_projects, number_of_rooms, number_of_periods
    End If

    Randomize
    r = Int(3 * Rnd) + 1
    If r = 1 Then
APPENDIX A: THE CODE

heuristic1_exam solution_exam, number_of_projects,
number_of_supervisors_examiners
ElseIf r = 2 Then
  heuristic2_exam solution_exam, number_of_projects,
number_of_supervisors_examiners
ElseIf r = 3 Then
  heuristic4_exam solution_exam, number_of_projects,
number_of_supervisors_examiners
End If

cost_new = Evaluate(solution(), solution_exam(),
number_of_projects, number_of_rooms, number_of_periods,
exams_per_period(), availability_of_rooms(),
suitability_of_projects_for_examiners(), availability_of_examiners())

If cost_new <= cost Then
  best_iter = 0
  cost = cost_new
  copyFromTo solution, solution_prev, number_of_projects,
number_of_rooms, number_of_periods
  copyFromTo_exam solution_exam, solution_exam_prev,
number_of_projects

  If cost < cost_best Then
    cost_best = cost_new
    copyFromTo solution, solution_best, number_of_projects,
number_of_rooms, number_of_periods
    copyFromTo_exam solution_exam, solution_exam_best,
number_of_projects
  End If

Else

  best_iter = best_iter + 1
  If best_iter = 1000 Then
    best_iter = 0
    cost = cost_new
    copyFromTo solution, solution_prev, number_of_projects,
number_of_rooms, number_of_periods
    copyFromTo_exam solution_exam, solution_exam_prev,
number_of_projects
  Else
    copyFromTo solution, solution_prev, number_of_projects,
number_of_rooms, number_of_periods
  End If
APPENDIX A: THE CODE

copyFromTo_exam solution_exam_prev, solution_exam, number_of_projects
    End If
    End If
Next iter

output solution_best, solution_exam_best, names_of_rooms, date_and_time_of_periods, _

CreatePanel = True
End Function

Sub copyFromTo(solution() As Integer, solution_prev() As Integer, number_of_projects As Integer, number_of_rooms As Integer, number_of_periods As Integer)
    Dim i As Integer, j As Integer, k As Integer
    For i = 1 To number_of_project
        For j = 1 To number_of_rooms
            For k = 1 To number_of_periods
                solution_prev(i, j, k) = solution(i, j, k)
            Next k
        Next j
    Next i
End Sub

Sub copyFromTo_exam(solution_exam() As Integer, solution_exam_prev() As Integer, number_of_projects As Integer)
    Dim i As Integer, j As Integer
    For i = 1 To number_of_projects
        For j = 1 To 3
            solution_exam_prev(i, j) = solution_exam(i, j)
        Next j
    Next i
End Sub

Function SheetExists(SheetName As String, wb As Excel.Workbook)
    Dim s As Excel.Worksheet
    On Error Resume Next

Set s = wb.Sheets(SheetName)
On Error GoTo 0
SheetExists = Not s Is Nothing
End Function

Function Evaluate(solution() As Integer, solution_exam() As Integer,
number_of_projects As Integer, number_of_rooms As Integer,
number_of_periods As Integer, exams_per_period() As Integer,
availability_of_rooms() As Integer,
suitability_of_projects_for_examiners() As Integer,
availability_of_examiners() As Integer) As Long
    Evaluate = C1(solution, number_of_projects, number_of_rooms,
number_of_periods) + C2(solution, number_of_projects, number_of_rooms,
number_of_periods) + _
    C3(solution, number_of_projects, number_of_rooms, number_of_periods,
exams_per_period) + C4(solution, number_of_projects, number_of_rooms,
number_of_periods, availability_of_rooms) + _
    C5(solution_exam, number_of_projects) + C6(solution, solution_exam,
number_of_projects, number_of_rooms, number_of_periods) + _
    C7(solution_exam, number_of_projects,
suitability_of_projects_for_examiners) + C8(solution, solution_exam,
number_of_projects, number_of_rooms, number_of_periods,
availability_of_examiners)
End Function

Sub init(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer)
    Dim i As Integer, j As Integer, k As Integer
    For i = 1 To proj
        For j = 1 To rooms
            For k = 1 To pds
                sol(i, j, k) = Int(2 * Rnd)
            Next k
        Next j
    Next i
End Sub

Sub init2(sol_ex() As Integer, proj As Integer, no_sups_ex As Integer)
    Dim i As Integer, j As Integer
    For i = 1 To proj
        For j = 1 To 3
APPENDIX A: THE CODE

sol_ex(i, j) = Int(no_sups_ex * Rnd) + 1
Next j
Next i
End Sub

'LLH1
'change assignment for solution representation 1
Sub heuristic1(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer)
    Dim i_rand As Integer, j_rand As Integer, k_rand As Integer
    Randomize
    i_rand = Int(proj * Rnd) + 1
    Randomize
    j_rand = Int(rooms * Rnd) + 1
    Randomize
    k_rand = Int(pds * Rnd) + 1

    If sol(i_rand, j_rand, k_rand) = 1 Then
        sol(i_rand, j_rand, k_rand) = 0
    Else
        sol(i_rand, j_rand, k_rand) = 1
    End If
End Sub

'LLH2
'change assignment for solution representation 1
Sub heuristic2(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer)
    Dim i_rand As Integer, j_rand As Integer, k_rand As Integer
    Dim i As Integer

    For i = 1 To 2
        Randomize
        i_rand = Int(proj * Rnd) + 1
        Randomize
        j_rand = Int(rooms * Rnd) + 1
        Randomize
        k_rand = Int(pds * Rnd) + 1
    Next i

APPENDIX A: THE CODE

If sol(i_rand, j_rand, k_rand) = 1 Then
    sol(i_rand, j_rand, k_rand) = 0
Else
    sol(i_rand, j_rand, k_rand) = 1
End If
Next i
End Sub

'LLH3
'unassign for solution representation 1
Sub heuristic3(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer)
    Dim i_rand As Integer, j_rand As Integer, k_rand As Integer
    Dim i As Integer
    For i = 1 To 5
        Randomize
        i_rand = Int(proj * Rnd) + 1
        Randomize
        j_rand = Int(rooms * Rnd) + 1
        Randomize
        k_rand = Int(pds * Rnd) + 1

        If sol(i_rand, j_rand, k_rand) = 1 Then
            sol(i_rand, j_rand, k_rand) = 0
        End If
    Next i
End Sub

'LLH4
'change assignment for soltion representation 2
Sub heuristic1_exam(sol_exam() As Integer, proj As Integer, examiners As Integer)
    Dim i_rand As Integer, j_rand As Integer
    Randomize
    i_rand = Int(proj * Rnd) + 1
    Randomize
    j_rand = Int(3 * Rnd) + 1

APPENDIX A: THE CODE

```vba
sol_exam(i_rand, j_rand) = Int(examiners * Rnd) + 1
End Sub

'LLH5
'change assignment for solution representation 2
Sub heuristic2_exam(sol_exam() As Integer, proj As Integer, examiners As Integer)
    Dim i_rand As Integer, j_rand As Integer
    Dim i As Integer

    For i = 1 To 2
        Randomize
        i_rand = Int(proj * Rnd) + 1
        Randomize
        j_rand = Int(3 * Rnd) + 1

        sol_exam(i_rand, j_rand) = Int(examiners * Rnd) + 1
    Next i
End Sub

'LLH6
Sub heuristic4_exam(sol_exam() As Integer, proj As Integer, examiners As Integer)
    Dim i_rand As Integer, j_rand As Integer
    Dim i_rand2 As Integer, j_rand2 As Integer
    Dim i As Integer
    Dim temp As Integer

    i_rand = Int(proj * Rnd) + 1
    j_rand = Int(3 * Rnd) + 1

    temp = sol_exam(i_rand, j_rand)

    i_rand2 = Int(proj * Rnd) + 1
    j_rand2 = Int(3 * Rnd) + 1
    sol_exam(i_rand, j_rand) = sol_exam(i_rand2, j_rand2)
    sol_exam(i_rand2, j_rand2) = temp
```
End Sub

' constraint 1: all projects must be assigned to timeslots
Function C1(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer
    Dim is_assigned As Boolean
    Dim cost As Long
    cost = 0
    For i = 1 To proj
        is_assigned = False
        For j = 1 To rooms
            For k = 1 To pds
                If sol(i, j, k) = 1 Then
                    is_assigned = True
                End If
            Next k
        Next j
        If is_assigned = False Then
            cost = cost + 1
        End If
    Next i
    C1 = cost
End Function

' constraint 2: only one timesalot must be assigned to the project
Function C2(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer
    Dim cost As Long
    Dim sum As Integer
    cost = 0
    For i = 1 To proj
        sum = 0
        For j = 1 To rooms
APPENDIX A: THE CODE

For k = 1 To pds
    sum = sum + sol(i, j, k)
Next k
Next j
If sum > 1 Then
    cost = cost + sum - 1
End If
Next i
C2 = cost
End Function

'constraint 3: max number of projects assigned to one period
Function C3(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer, no_of_exams() As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer
    Dim is_valid As Boolean
    Dim cost As Long
    cost = 0
    Dim sum As Integer
    sum = 0
    For j = 1 To rooms
        For k = 1 To pds
            is_valid = False
            For i = 1 To proj
                sum = sum + sol(i, j, k)
            Next i
            If sum = no_of_exams(k) Then
                is_valid = True
            End If
            If is_valid = False Then
                cost = cost + 1
            End If
        Next k
    Next j
    C3 = cost
APPENDIX A: THE CODE

End Function

' constraint 4: Unavailable rooms must not be assigned
Function C4(sol() As Integer, proj As Integer, rooms As Integer, pds As Integer, rooms_avail() As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer
    Dim is_valid As Boolean
    Dim cost As Long
    cost = 0
    For i = 1 To proj
        is_valid = True
        For j = 1 To rooms
            For k = 1 To pds
                If sol(i, j, k) = 1 And rooms_avail(j, k) = 0 Then
                    is_valid = False
                    cost = cost + 1
                End If
            Next k
        Next j
    Next i
    C4 = cost
End Function

' constraint 5: the three committee members can not be the same examiner
Function C5(sol1() As Integer, proj As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer
    Dim cost As Long
    cost = 0
    For i = 1 To proj
        For j = 1 To 3
            For k = 1 To j - 1
                If sol1(i, j) = sol1(i, k) Then
                    cost = cost + 1
                End If
            Next k
        Next j
    Next i
    C5 = cost
End Function
APPENDIX A: THE CODE

End If
Next k
Next j
Next i
C5 = cost
End Function

' constraint 6 : The examiner must not be assigned to two projects on different rooms at the same period
Function C6(sol() As Integer, sol1() As Integer, proj As Integer, rooms As Integer, pds As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer, l As Integer, m As Integer
    Dim current_examiner As Integer
    Dim current_session As Integer
    Dim cost As Long
    cost = 0

    For i = 1 To proj
        For j = 1 To 3
            current_examiner = sol1(i, j)
            current_session = -1
            For k = 1 To rooms
                For l = 1 To pds
                    If sol(i, k, l) = 1 Then
                        current_session = l
                        Exit For
                    End If
                Next l
                If current_session <> -1 Then
                    Exit For
                End If
            Next k
            For k = 1 To proj
                If i <> k Then
                    For l = 1 To rooms

APPENDIX A: THE CODE

For m = 1 To pds
    If sol(k, 1, m) = 1 And current_session = m Then
        For n = 1 To 3
            If sol1(k, n) = current_examiner Then
                cost = cost + 1
            End If
        Next n
    End If
Next m
Next l
End If
Next k
Next j
Next i
C6 = cost
End Function

' constraint 7 : Assigning an examiner to a project panel that he choose not to be in
Function C7(sol1() As Integer, proj As Integer, exam_suit() As Integer) As Long
    Dim i As Integer, j As Integer
    Dim cost As Long
    cost = 0
    For i = 1 To proj
        For j = 1 To 3
            If exam_suit(i, sol1(i, j)) = 2 Then
                cost = cost + 1
            End If
        Next j
    Next i
C7 = cost
End Function

' constraint 8 : Assigning an examiner to a timeslot that he is not available at
Function C8(sol() As Integer, sol1() As Integer, proj As Integer, rooms As Integer, pds As Integer, exam_avail() As Integer) As Long
    Dim i As Integer, j As Integer, k As Integer, l As Integer
    Dim current_session As Integer
    Dim cost As Long
    cost = 0
    For i = 1 To proj
        For j = 1 To rooms
            For k = 1 To pds
                If sol(i, j, k) = 1 Then
                    current_session = k
                    For l = 1 To 3
                        If exam_avail(sol1(i, l), k) = 0 Then
                            cost = cost + 1
                        End If
                    Next l
                    Exit For
                End If
            Next k
        Next j
    Next i
    C8 = cost
End Function
## APPENDIX B: THE INPUT FILES

### Table B-1: Supervisors_Examinersheet

<table>
<thead>
<tr>
<th>المنتمون والمشاركون</th>
</tr>
</thead>
<tbody>
<tr>
<td>أ.د. مصطفى نواري</td>
</tr>
<tr>
<td>د. ميرغني فتح الرحمن</td>
</tr>
<tr>
<td>أ. عزة الجيلي</td>
</tr>
<tr>
<td>أ.د. شريف فضل بابكر</td>
</tr>
<tr>
<td>أ. رانيا نعم</td>
</tr>
<tr>
<td>أ. جهان الطيب</td>
</tr>
<tr>
<td>أ.د. سامي محمد شريف</td>
</tr>
<tr>
<td>د. حجاج هاشم</td>
</tr>
<tr>
<td>أ. غيسي يحيا</td>
</tr>
<tr>
<td>د. إمام أبومعالي</td>
</tr>
<tr>
<td>د. محمد علي</td>
</tr>
<tr>
<td>أ. محمد النوجري</td>
</tr>
<tr>
<td>د. كمال رمضان</td>
</tr>
<tr>
<td>د. الأمين حمدة</td>
</tr>
<tr>
<td>د. الفاضل زكريا</td>
</tr>
<tr>
<td>أ. مهند أحمد محمد</td>
</tr>
<tr>
<td>د. حميد عباس</td>
</tr>
<tr>
<td>د. أحمد عبد جбри</td>
</tr>
<tr>
<td>أ. عزة حسن</td>
</tr>
<tr>
<td>د. عبد الرحمن كرار</td>
</tr>
<tr>
<td>أ. طارق أحمد خالد</td>
</tr>
</tbody>
</table>
## APPENDIX B: THE INPUT FILES

### Table B-2: Projects

<table>
<thead>
<tr>
<th>المرفوع</th>
<th>اسم المشروع</th>
<th>الطلاب</th>
</tr>
</thead>
<tbody>
<tr>
<td>آ.د. مصطفى نواري</td>
<td>Power System Fault stability simulator</td>
<td>أ.د. مصطفى نواري</td>
</tr>
<tr>
<td>آ.د. مصطفى نواري</td>
<td>Power System Stability Investigation</td>
<td>آ.د. مصطفى نواري</td>
</tr>
<tr>
<td>آ.د. مصطفى نواري</td>
<td>Power System Control</td>
<td>آ.د. مصطفى نواري</td>
</tr>
<tr>
<td>آ.د. مصطفى نواري</td>
<td>Aircraft lateral control</td>
<td>آ.د. مصطفى نواري</td>
</tr>
<tr>
<td>آ.د. مصطفى نواري</td>
<td>Simulation of guidance missile</td>
<td>آ.د. مصطفى نواري</td>
</tr>
<tr>
<td>آ.د. مصطفى نواري</td>
<td>Maximizing Solar Energy Input For Cubesat Using Solar Tracker And A Maximum Power Point Tracker</td>
<td>آ.د. مصطفى نواري</td>
</tr>
<tr>
<td>آ.د. بشر فضل باكر</td>
<td>Design and Development of Circuit Diagnosis Tool for Prepaid Energy Meter</td>
<td>آ.د. بشر فضل باكر</td>
</tr>
<tr>
<td>آ.د. بشر فضل باكر</td>
<td>Design and Implementation of Electric Vehicle</td>
<td>آ.د. بشر فضل باكر</td>
</tr>
<tr>
<td>آ.د. بشر فضل باكر</td>
<td>Design of CNC machine</td>
<td>آ.د. بشر فضل باكر</td>
</tr>
<tr>
<td>آ.د. بشر فضل باكر</td>
<td>ECG Signal Analysis</td>
<td>آ.د. بشر فضل باكر</td>
</tr>
<tr>
<td>آ.د. بشر فضل باكر</td>
<td>Eye Controlled Mouse</td>
<td>آ.د. بشر فضل باكر</td>
</tr>
<tr>
<td>آ.د. بشر فضل باكر</td>
<td>3D mapping Robot for Homes</td>
<td>آ.د. بشر فضل باكر</td>
</tr>
<tr>
<td>آ.د. ساري صديق عبدالله</td>
<td>Hand Controlled Virtual Reality Electronics Laboratory</td>
<td>آ.د. ساري صديق عبدالله</td>
</tr>
<tr>
<td>آ.د. ساري صديق عبدالله</td>
<td>MPLS design and simulation using GNS3</td>
<td>آ.د. ساري صديق عبدالله</td>
</tr>
<tr>
<td>د. سماح إبراهيم</td>
<td>Design and implementation of FPGA-based pipelined</td>
<td>د. سماح إبراهيم</td>
</tr>
</tbody>
</table>

B-2
**APPENDIX B: THE INPUT FILES**

<table>
<thead>
<tr>
<th>Project Title</th>
<th>NAME</th>
<th>University</th>
</tr>
</thead>
<tbody>
<tr>
<td>microprocessor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vision PID controlled ball on blade stabilizer</td>
<td>عمر</td>
<td>إبراهيم</td>
</tr>
<tr>
<td>Internet of Things based on Arduino</td>
<td>محمد</td>
<td>عبد الله</td>
</tr>
<tr>
<td>Speech analysis and synthesis using LPC and VAD</td>
<td>عمر</td>
<td>مصطفى</td>
</tr>
<tr>
<td>Musical Notes Recognition Application</td>
<td>رزان</td>
<td>عبد الله</td>
</tr>
<tr>
<td>Internet Based Monitoring Using Arduino &amp; GSM Shield Kit</td>
<td>عمر</td>
<td>عبد الله</td>
</tr>
<tr>
<td>3D printer design and implementation using arduino kit</td>
<td>أ.إب</td>
<td>عبد الله</td>
</tr>
<tr>
<td>Fingerprint recognition using neural networks</td>
<td>ليث</td>
<td>عبد الله</td>
</tr>
<tr>
<td>Session initiation protocol implementation using android enviroment</td>
<td>هبة</td>
<td>عبد الرحمن</td>
</tr>
<tr>
<td>Fall detection system based on android on a smartphone (HEEDER)</td>
<td>سامح</td>
<td>عثمان</td>
</tr>
<tr>
<td>MIMO and OFDMA implementation in LTE networks</td>
<td>سارة</td>
<td>عبد الله</td>
</tr>
<tr>
<td>Adaptive Modulation and Dynamic Bandwidth in LTE Systems</td>
<td>محمد</td>
<td>أحمد</td>
</tr>
<tr>
<td>Channel coding in mobile networks</td>
<td>عفاف</td>
<td>عبد الرحمن</td>
</tr>
<tr>
<td>Channel Modelling and performance evaluation for High Altitude Platforms (HAPs)</td>
<td>ملاك</td>
<td>عبد الله</td>
</tr>
<tr>
<td>QoS Performance Analysis for Voice over LTE Mobile Networks</td>
<td>شهيد</td>
<td>عبد الله</td>
</tr>
</tbody>
</table>

**B-2**
<table>
<thead>
<tr>
<th><strong>APPENDIX B: THE INPUT FILES</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Analysis of Mobility Management in LTE Systems</strong></td>
</tr>
<tr>
<td><strong>Investigation of small home PV Grid connection (case study)</strong></td>
</tr>
<tr>
<td><strong>Series and Shunt Compensation for long transmission lines (Case Study: Atbara-Port Sudan transmission line)</strong></td>
</tr>
<tr>
<td><strong>Harmonics mitigation and power factor correction in industrial systems</strong></td>
</tr>
<tr>
<td><strong>Study of Protection at Khartoum North power Station</strong></td>
</tr>
<tr>
<td><strong>Starting of electrical motors with a different approach on starting DC motors</strong></td>
</tr>
<tr>
<td><strong>Interfacing of Three Phase Grid Connected Photovoltaic System</strong></td>
</tr>
<tr>
<td><strong>Design, Implementation and Control of a Quadcopter</strong></td>
</tr>
<tr>
<td><strong>Reconfigurable Composable Conveyor System (RCCS)</strong></td>
</tr>
<tr>
<td><strong>Power factor Correction using Power Electronics</strong></td>
</tr>
<tr>
<td><strong>DC-Motor Speed Control using PID-Controller</strong></td>
</tr>
<tr>
<td><strong>Practical Micro Grid Design and Control using Multi-Agent System for Shambat Distribution Network</strong></td>
</tr>
<tr>
<td><strong>3D reconstruction from 2D images</strong></td>
</tr>
<tr>
<td><strong>Solving Traveling Salesman Problem (TSP) Worldwide Using Hyper Heuristic</strong></td>
</tr>
</tbody>
</table>
# APPENDIX B: THE INPUT FILES

<table>
<thead>
<tr>
<th>Author</th>
<th>Title</th>
<th>Title (Arabic)</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.</td>
<td>Solving timetabling problems using hyper-heuristics</td>
<td>عيدالعزيز محمود - فداء عوض طه محمد - إنين محمود</td>
</tr>
<tr>
<td>A. طارق أحمد عادل</td>
<td>Anti-money Laundering Software</td>
<td>يس ابراهيم احمد الظاهر - منذر محمد عل &quot;محمد</td>
</tr>
<tr>
<td>A. طارق أحمد عادل</td>
<td>Caffe-Based Implementation and Benchmark Analysis of Popular Visual Recognition Deep CNN Architectures</td>
<td>أحمد عبدالعظيم احمد بابكر - مصطفى الغالي السيد محمد</td>
</tr>
<tr>
<td>A. طارق أحمد عادل</td>
<td>Sentement Analysis For Social Media</td>
<td>عمر محمد الفلاح موسي محمد - أحمد عبدالصالح محمد</td>
</tr>
</tbody>
</table>
# APPENDIX B: THE INPUT FILES

Table B-3: rooms

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
</tr>
<tr>
<td></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
</tr>
<tr>
<td></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
</tr>
<tr>
<td></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
</tr>
<tr>
<td></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
<td><em><strong>1</strong></em></td>
</tr>
<tr>
<td></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
<td><em><strong>0</strong></em></td>
</tr>
</tbody>
</table>

*‘1’ if available, ‘0’ otherwise*
APPENDIX B: THE INPUT FILES

Table B-4: Sample of suitability sheet

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>2</th>
<th>2</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power System Fault stability simulator</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Power System Control</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Aircraft lateral control</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Simulation of guidance missile</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Maximizing Solar Energy Input For Cubesat Using Solar Tracker And A Maximum Power Point Tracker</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Design and Development of Circuit Diagnosis Tool for Prepaid Energy Meter</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table B-5: Availability sheet

<table>
<thead>
<tr>
<th></th>
<th>11:30–14:30</th>
<th>8:30–11:30</th>
</tr>
</thead>
<tbody>
<tr>
<td>المسمى</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>المسمى</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>المسمى</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>المسمى</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>المسمى</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>اليوم</td>
<td></td>
<td></td>
</tr>
<tr>
<td>الوقت</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## APPENDIX B: THE INPUT FILES

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- د. ميغفي فتح الرحمن
- د. عزة الجيلي
- د. شريف فضل بايكر
- د. رانيا نعيم
- د. وح悯ان الطيب
- د. سامي محمود شريف
- د. سماح ىاشم
- د. سماح ىاشم
- د. عيسى يحيا
- د. إيمان ابوالدعا
- د. محمد علي
- د. محمد النوراوي
- د. كمال رمضان
- د. الأمين حمودة
- د. الفاضل زكرياء
- د. مهند أحمد محمد
- د. خالد عباس
- د. أحمد عبد حورى
- د. عزة حسن
- د. عبد الرحمن كرار
- د. طارق أحمد حلال
## APPENDIX C: RESULTS SAMPLE

### Table C-1: Sample of the result

<table>
<thead>
<tr>
<th>Enrollment 1</th>
<th>Enrollment 2</th>
<th>Enrollment 3</th>
<th>Module</th>
<th>Room</th>
<th>Day</th>
<th>Time</th>
<th>Supervisor</th>
<th>Project Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Lecture</td>
<td>Tuesday</td>
<td>8:30 - 11:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Power System Fault stability simulator</td>
</tr>
<tr>
<td>A. Mr. Ahmed Khalid</td>
<td>Lecture</td>
<td>Wednesday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Power System Stability Investigation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>8:30 - 11:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Power System Control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Aircraft lateral control</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Maximizing Solar Energy Input For Cubesat Using Solar Tracker And A Maximum Power Point Tracker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>8:30 - 11:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Design and Development of Circuit Diagnosis Tool for Prepaid Energy Meter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Design and Implementation of Electric Vehicle</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>8:30 - 11:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Design of CNC machine</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>ECG Signal Analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>Eye Controlled Mouse</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A. Mr. Ahmed Shafiq Fath Mohd</td>
<td>Lecture</td>
<td>Thursday</td>
<td>11:30 - 14:30</td>
<td>A. Dr. M Mostafa Nawar</td>
<td>3D mapping Robot for Homes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## APPENDIX C: RESULTS SAMPLE

<table>
<thead>
<tr>
<th>Time</th>
<th>Venue</th>
<th>Course Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Hand Controlled Virtual Reality Electronics Laboratory</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Design and implementation of FPGA-based pipelined microprocessor</td>
</tr>
<tr>
<td>8:30 - 11:30</td>
<td>Hall 12-1</td>
<td>Vision PID controlled ball on plate stabilizer</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Internet of Things based on Arduino</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Speech analysis and synthesis using LPC and VAD</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Musical Notes Recognition Application</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Internet Based Monitoring Using Arduino &amp; GSM Shield Kit</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>3D printer design and implementation using arduino kit</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Fingerprint recognition using neural networks</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>Session initiation protocol implementation using android environment</td>
</tr>
<tr>
<td>8:30 - 11:30</td>
<td>Hall 12-1</td>
<td>Fall detection system based on android on a smartphone (HEEDER)</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Hall 12-1</td>
<td>MIMO and OFDMA implementation in LTE networks</td>
</tr>
<tr>
<td>Time</td>
<td>Day</td>
<td>Lecture Topic</td>
</tr>
<tr>
<td>--------------</td>
<td>--------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Adaptive Modulation and Dynamic Bandwidth in LTE Systems</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Channel coding in mobile networks</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Channel Modelling and performance evaluation for High Altitude Platforms (HAPs)</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>QoS Performance Analysis for Voice over LTE Mobile Networks</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Analysis of Mobility Management in LTE Systems</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Investigation of small home PV Grid connection (case study)</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Series and Shunt Compensation for long transmission lines (Case Study: Atbara-Port Sudan transmission line)</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Harmonics mitigation and power factor correction in industrial systems</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Study of Protection at Khartoum North power Station</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Starting of electrical motors with a different approach on starting DC motors</td>
</tr>
<tr>
<td>11:30 - 14:30</td>
<td>Tuesday</td>
<td>Interfacing of Three Phase Grid Connected Photovoltaic System</td>
</tr>
</tbody>
</table>
### APPENDIX C: RESULTS SAMPLE

<table>
<thead>
<tr>
<th>Time</th>
<th>Course Title</th>
<th>Presenter(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:30-11:30</td>
<td>Design, Implementation and Control of a Quadcopter</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Reconfigurable Composable Conveyor System (RCCS)</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Power factor Correction using Power Electronics</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>8:30-11:30</td>
<td>DC-Motor Speed Control using PID-Controller</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Practical Micro Grid Design and Control using Multi-Agent System for Shambat Distribution Network</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>3D reconstruction from 2D images</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Solving Traveling Salesman Problem (TSP) Worldwide Using Hyper Heuristic</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Solving timetabling problems using hyper-heuristics</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Anti-money Laundering Software</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Caffe-Based Implementation and Benchmark Analysis of Popular Visual Recognition Deep CNN Architectures</td>
<td>د. شريف فضل بابكر</td>
</tr>
<tr>
<td>11:30-14:30</td>
<td>Sentiment Analysis For Social Media</td>
<td>د. شريف فضل بابكر</td>
</tr>
</tbody>
</table>

---

### APPENDIX B: THE INPUT FILES

#### Design, Implementation and Control of a Quadcopter

- **د. شريف فضل بابكر**
- **الخميس 1-12**
- **8:30-11:30**
- **قاعة 121-1-12**

#### Reconfigurable Composable Conveyor System (RCCS)

- **د. أحمد حسن**
- **الاحد 11-30**
- **11:30-14:30**
- **قاعة 121-1-12**

#### Power factor Correction using Power Electronics

- **د. أحمد حسن**
- **الأربعاء 11-30**
- **11:30-14:30**
- **قاعة 121-1-12**

#### DC-Motor Speed Control using PID-Controller

- **د. أحمد حسن**
- **الخميس 1-12**
- **8:30-11:30**
- **قاعة 121-1-12**
Our project provide an easy user interface. When a user wants to use the program a button named استخراج جدول المناقشات is pressed, the next window appears:

![Figure D-1: shows user interface](image)

After the input file name is written in a right way the program starts to work and output it as shown in appendix C.