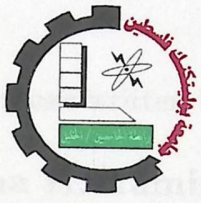
Palestine Polytechnic University

College of Administrative Sciences and Informatics
Information Technology Department *

College of Applied Sciences
Mathematics and Computer Science Department **



University Course Scheduling Using Parallel Multi-objective Evolutionary Algorithms

Prepared by:

Insaf K. Al-Najjar *

Muna H. Tamimi *

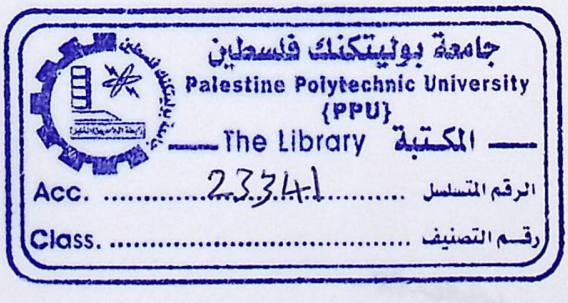
Tareq O. Takruri **

Supervised by:

Dr. Mohammad Aldasht *

This project submitted in partial fulfilment of the requirements for the degree B.Sc.

June, 2009



Palestine Polytechnic University
Hebron - Palestine

University Course Scheduling Using Parallel Multi-objective Evolutionary Algorithms

A thesis written by

Tareq O. Takruri Muna H. Tamimi Insaf K. Al-Najar

Under the project supervisor guidance and with the approval of all members of the board of examiners, the project was submitted to the Information Technology Department at the College of Administrative Science and Informatics and Mathematics and Computer Science Department at the College of Applied Science to meet the requirements of the B.Sc of Information Technology and B.Sc of Computer Science.

Supervisor Name : ..... Supervisor Signed : .....

Dept.Chairperson Name: ..... Dept.Chairperson Signed: .....

06-2009

Abstract

Evolutionary Algorithm (EA) provides a mechanism that can achieve efficient exploration for design spaces. Thus, it constitutes an efficient tool for identifying the best alternatives to implement the solution of a certain problem. In a previous work of Information Technology students, they have applied the EA to the university course scheduling problem and they have implemented the methodology on a real data from the College of Administrative Sciences and Informatics at Palestine Polytechnic University (PPU). Two major shortages were founded in their project: First, the relatively long execution time that takes the evolutionary algorithm to find the optimal solution. Second, the soft constraints were considered in the implementation, but were not well satisfied.

In this project, we have implemented the EA using parallel programming techniques. This permits to execute the program in a cluster of machines, which in turns reduces the execution time. In addition, we have applied some soft constraints concurrently with the hard constraints in order to get better results by making the EA minimizing the soft cost without affecting the hard cost.

Results show that, after redrafting the algorithm to be multi-objective, the soft cost will go to zero if we use enough individuals and iterations, at the same time the hard constraints are still satisfied.

In addition, after distributing the algorithm on 7 machines with 11 processors the obtained speedup reaches 6 on average.

ملخص

يقوم هذا البحث على إيجاد حل مناسب في فضاء مشكلة جدولة مواعيد محاضرات الجامعة عن طريق تصميم خوارزمية تطورية (Evolutionary Algorithm)، حيث تراعي الأمور المتبعة في حل المشكلة وإيجاد حل قريب من الحل المثالي.

يقوم البحث بإنشاء برنامج لمساقات الجامعة بحيث يراعي القيود الموضوعة بنوعيها (hard, soft)، فينتج لدينا تعيين لشعب المساقات في وقت محدد وقاعة مناسبة بحيث لا يوجد أي تعارض زمني مع أي مساق أخر .

في عمل سابق لطلاب دائرة تكنلوجيا المعلومات، تم تطبيق خوارزمية تطورية على بيانات حقيقية من كلية العلوم الإدارية ونظم المعلومات في جامعة بوليتكنيك فلسطين لحل المشكلة، وقد تم التوصل لنتائج مرضية، إلا ان الحل واجهته بعض النواقص، الأول هو الفترة الزمنية الطويلة التي تستغرقها الخوارزمية لإيجاد الحل المثالي. والثاني هو ان بعد أخذ (soft constraints) بعين الإعتبار لم يكن الحل مرضياً.

في هذا المشروع، قمنا بتطبيق الخوارزمية السابق ذكرها بإستخدام تقنيات البرمجة الموازية عن طريق مكتبة - MPI -. ذلك أتاح تشغيل النظام على مجموعة أجهزة موزعة، مما أدي بدورة لتقليل الوقت التنفيذي للخوارزمية. من ناحية أخرى، قمنا بإضافة بعض (Soft Constraints) بالتوازي مع (Hard Constraints) على الخوارزمية للحصول على نتائج أفضل وتقليل أوقات الفراغ بين المحاضرات .

تبين النتائج انه بعد إعادة صياغة الخوارزمية لتصبح موازية ومتعددة الأهداف، تم تقليل (soft cost) لتصبح قيمته تؤول الى الصفر دون التاثير على (hard cost). كما تم تسريع تنفيذ الخوارزمية بنسبة 6 اضعاف لما كانت عليه وذلك بعد توزيعه على 7 اجهزة.

Acknowledgements

We would like to thank our teachers who help, advice and teach us to write this documintation and give us the research skills, Dr. Mohammad Aldasht and Dr. Mahmoud Alsaheb.

UCS using PMEA team

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

University Course Scheduling (UCS) can be considered as an instant of what so called timetabling problem. In which time slots and teachers must be assigned to a set of courses in a way that satisfies a set of hard constraints and minimize the cost of another set of soft constraints [10]. This problem is considered to be a non-polynomial-time hard (NP-hard) problem, which means that the amount of computation required to find solutions increases exponentially with problem size [18].

NP-hard, a problem H is NP-hard if and only if there is an NP-complete problem L that is polynomial time Turing-reducible to H, NP-hard problem may be of any type: decision problems, search problems, optimization problems [3].

1.2 Abbreviations

- Palestinian Polytechnic University, *PPU*
- Evolutionary Algorithm, *EA*
- University Course Scheduling, *UCS*
- Message Passing Interface, *MPI*

- Parallel Evolutionary Algorithm, *PEA*
- Multi-objective Parallel Evolutionary Algorithm, *MPEA*
- Single Program Multiple Data, *SPMD*
- Genetic Algorithm, *GA*
- Exploitation of the Fastest Processor, *EFP*
- Gene Expression programming client, *GEP-client*
- Optimization client, *O-client*
- Network File System, *NFS*
- Secure shell, *SSH*
- Integrated development Environment, *IDE*
- Nondeterministic Polynomial-time hard problem, *NP hard problem*
- Multi-objective optimization, *MOOP*
- Traveling Salesman Problem, *TSP*
- Evolution Strategies, *ES*
- Genetic Programming, *GP*
- Evolutionary Programming, *EP*

1.3 Project objectives

This project is expected to achieve the following objectives:

- Enhance the algorithm solution.
- To speedup the algorithm.

- To enhance the design and development of the approach by using object oriented programming.
- To show that the parallel multi-objective algorithm is better than the sequential single-objective one.

1.4 Scheduling problem

Scheduling problem, in general, can be defined as a problem of finding the optimal sequence for executing a finite set of operations under a set of certain constraints [15]. Scheduling problem has several types, such as: multiprocessor scheduling, shop scheduling, staff scheduling, timetable design, etc. In this work, the university course scheduling problem will be handled; this includes, the process of assigning a set of classrooms and set of instructors to a given set of courses taking into account a set of constraints.

The final solution is required to satisfy a set of constraints, these constraints are divided into hard constraints, which must be satisfied (have infinity cost), and soft constraints which should be satisfied (have less cost compared to hard constraints). Examples of hard constraints are: no person can be in more than one place at a time, and the total resources allocated to some time slot must be less than or equal to the resources that are available in that period, etc. Examples of soft constraints are: some teacher should not have very late classes daily, a group of students should not have consecutive classes in different and large distant places, and other individuals preferences, etc.

1.5 University course scheduling stages

The aim of this project is to solve the UCS problem at PPU by a parallel and multi-objective EA. This is a complex job, so it was divided into a set of stages as follows:

1.5.1 Sequential single-objective UCS

The first stage of the project was to find a number of trade-off solutions using an evolutionary algorithm. This phase has been done by the a previous graduation project [3]. The obtained solutions have been found much better than the manually solution, because they were performed in much less time and provided more alternatives to the students. Their algorithm took into account the prerequirements of the instructors, students, and locations as hard constraints, but they didn't take into account the soft constrains for both instructors and students, and the execution was on a single processor and took long time to return the results.

1.5.2 Sequential multi-objective UCS

The second phase aims to solve the single-objective UCS problem by taking both hard and soft constrains into consideration. Although the first phase led to a good result, it did not have enough satisfaction by the instructors and students, because sometimes the students have to wait for a long time between the lectures or they have to take many lectures without a break. These problems should be overcome in this phase of the project which makes it a multi-objective approach.

1.5.3 Parallel multi-objective UCS

The third phase of this project is to implement the algorithm in parallel programming technique, in order to make it faster. Message-Passing Interface (MPI) library is suitable for distributing such problems on a distributed memory parallel machines, to get benefit from the available network computing resources which lead to faster execution and higher utilization. This phase will improve the performance of scheduling because the problem is solved by more than one computer working in parallel rather than the sequential implementation of the first and second phase.

1.6 The organization of this document

This document is organized into the following chapters:

- Chapter 1: this chapter introduces the UCS definition, the problem that must be solved using EA, the objectives of our project, and the stages that the UCS passes through in order to be a complete and efficient solution.
- Chapter 2: this chapter aims to provide a clear view on strategies that can be used to solve the UCS problem, it contains demonstration about the evolutionary search methods with a flowchart and a pseudo-code for EA, also it represent a clear view about multi-objective optimization and parallel algorithms, Finally it contains the related projects and researches which its goal was to solve the parallel multi-objective UCS problem.
- Chapter 3: in this chapter, the details of the parallel multi-objective UCS problem description are explained, in term of defining the sets of the problem, how to solve it, the constraints that are taken into consideration during solving the problem. Then an explanation about the implementation of the project is represented, followed by listing flowcharts for every function of the evolutionary algorithm that have been used.
- Chapter 4: in this chapter the experimental results have been viewed.
- Chapter 5: in this chapter, number of final results and future works that is desired to enhance the parallel multi-objective UCS have been mentioned.
- Appendix: finally, two appendices have been added to this documintation, the first for the software engineering issues and the second is the time table that we have achieved after implementing the algorithm on 700 individulas for 700 generations.

Chapter 2

Background

2.1 Overview

In this chapter we will present the previous works that are related to EA, then we will talk in details about the EA that is used to solve the UCS problem, multi-objective strategy which is performed to decrease the hard and soft constrains, and parallel programming techniques which will reduce the required execution time.

2.2 Theoretical background

In this section we will talk about the EA approach in general and the multi-objective EA in specific. After that, we will talk about the parallel algorithms:

2.2.1 Evolutionary algorithm

Evolutionary algorithm (EA) is a part of Artificial Intelligence field where the solution is inspired by the biological behavior in the mechanisms of evolution. Certain ideas such as survival and natural selection, mutation and group-work have contributed in the creation of the evolutionary algorithm [2].

EA starts with selecting a set of individuals, each one represents a solution to the problem. Elements of the whole population are mutated to produce children

who are added to the population. This doubles the population size. After that, a selection process is performed on the whole population based on their closeness to the optimal solution. Weak individuals are discarded and the whole process is repeated with the remaining individuals to produce the next generation. Figure 2.1 shows the process of EA.

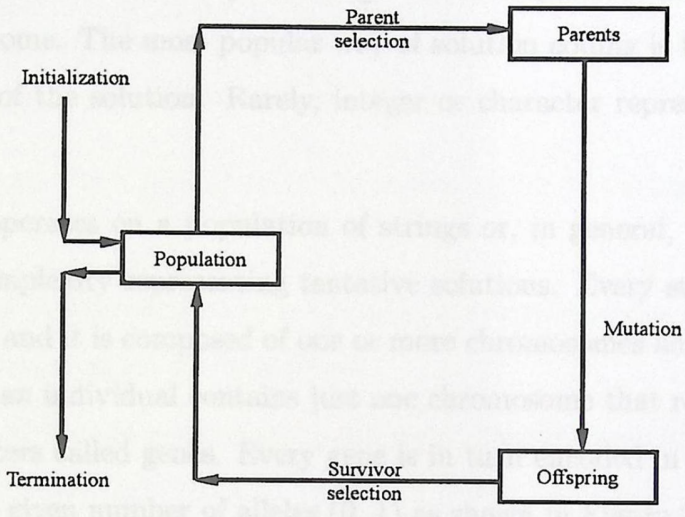


Figure 2.1: General Scheme of EAs

Several different types of evolutionary search methods were developed independently. These include:

1. Genetic algorithms: Genetic algorithms were invented by John H. Holland to follow the natural selection and evolution process [2]. In the nature, each species has to adapt itself according to a complex environmental conditions and changes with the purpose of maximizing the probability to survive. The knowledge collected by each species is coded in chromosomes, which go through transformations when they are reproduced.

   During a period of time those changes on the chromosomes will reproduce a species with greater probability of survive, and has a greater probability of passing its enhanced characteristics to future generations. Surly, not all

changes are beneficial, thus, those whose changes are not beneficial tend to die.

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

The GA operates on a population of strings or, in general, structures of arbitrary complexity representing tentative solutions. Every string is called an individual and it is composed of one or more chromosomes and a fitness value. Normally, an individual contains just one chromosome that represents the set of parameters called genes. Every gene is in turn encoded in (usually) binary by using a given number of alleles (0, 1) as shown in Figure 2.2.

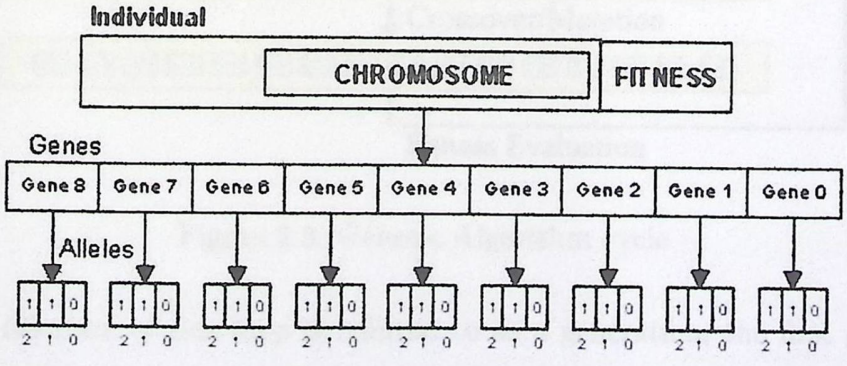


Figure 2.2: Individual main parts

An initial population of legal chromosomes is constructed randomly. The fitness of each chromosome is calculated every generation. A group of the

best chromosomes are then selected to produce the descendant of the next generation, which inherits the best characteristics from a pair of parents. After many generations of selecting the most suitable chromosomes, the result should be a population with a considerably better fitness than the original. GAs can be considered as an optimization techniques based on the concepts of natural and genetic selection [11] [14].

When the genetic algorithm is implemented it is done in a manner that involves the following cycle and shown in the Figure 2.3:

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

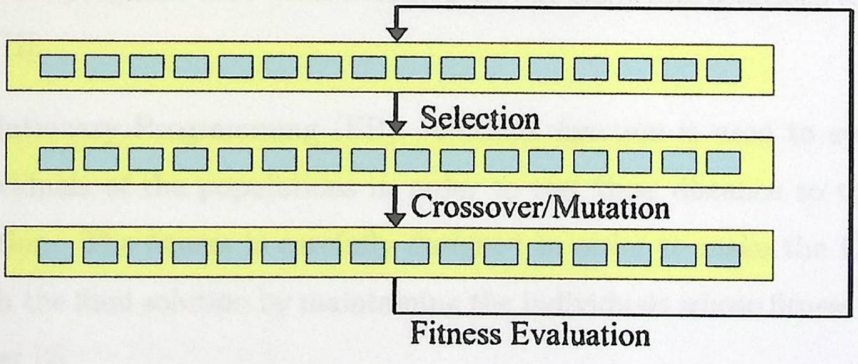


Figure 2.3: Genetic Algorithm cycle

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

2. Evolution Strategies (ES): ES is an optimization technique, Developed by Rechenberg (1965, 1973) and Schwefel (1965, 1977) at the Technical University

best chromosomes are then selected to produce the descendant of the next generation, which inherits the best characteristics from a pair of parents. After many generations of selecting the most suitable chromosomes, the result should be a population with a considerably better fitness than the original. GAs can be considered as an optimization techniques based on the concepts of natural and genetic selection [11] [14].

When the genetic algorithm is implemented it is done in a manner that involves the following cycle and shown in the Figure 2.3:

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

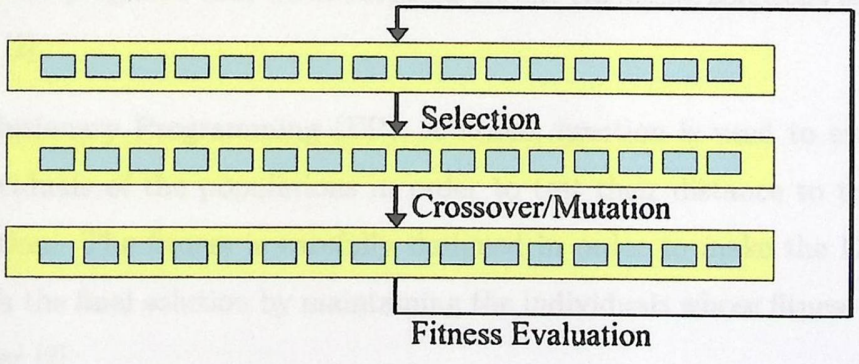


Figure 2.3: Genetic Algorithm cycle

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

2. Evolution Strategies (ES): ES is an optimization technique, Developed by Rechenberg (1965, 1973) and Schwefel (1965, 1977) at the Technical University

of Berlin, based on ideas of adaptation and evolution, which use mutation, recombination, and selection applied to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions. ESs have the following basic characteristics:

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

3. Genetic Programming (GP): GP is the extension of the genetic model of learning into the space of programs. It applies the evolutionary search principle to automatically develop computer programs in suitable languages (often LISP, but others are possible as well), genetic programming breeds computer programs. That means the objects that establish the population are not fixed-length character strings that encode possible solutions to the problem, but they are programs that when executed are the candidate solutions to the problem [2].

4. Evolutionary Programming (EP): A fitness function is used to evaluate the individuals of the populations in order to test their distance to the optimal solution. The fitness is carefully designed in order to make the EA process reach the final solution by maintaining the individuals whose fitness values are better [2].

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

of Berlin, based on ideas of adaptation and evolution, which use mutation, recombination, and selection applied to a population of individuals containing candidate solutions in order to evolve iteratively better and better solutions. ESs have the following basic characteristics:

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

3. Genetic Programming (GP): GP is the extension of the genetic model of learning into the space of programs. It applies the evolutionary search principle to automatically develop computer programs in suitable languages (often LISP, but others are possible as well), genetic programming breeds computer programs. That means the objects that establish the population are not fixed-length character strings that encode possible solutions to the problem, but they are programs that when executed are the candidate solutions to the problem [2].

4. Evolutionary Programming (EP): A fitness function is used to evaluate the individuals of the populations in order to test their distance to the optimal solution. The fitness is carefully designed in order to make the EA process reach the final solution by maintaining the individuals whose fitness values are better [2].

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

statistical distribution which weights minor variations in the behaviour of the offspring as highly probable and substantial variations. Further, the intensity of mutations is often reduced as the global optimum is approached. A special concern with the mutation, when the global optimum is not already known, several techniques have been proposed and implemented which address this difficulty, the most widely studied being the "Meta-Evolutionary" technique in which the variance of the mutation distribution is subject to mutation by a fixed variance mutation operator and evolves along with the solution [2].

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

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

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

(a) Initialize the population.

(b) Evaluate individuals.

(c) Randomly mutate each parent population member.

(d) Select members of new population.

(e) Go to step b until some condition is met.

(f) Evaluate individuals.

The following is a description of each step of the EA:

Step 1: Initialize the population This step depends on the problem that to be solved using EA, usually it contains the preparation of the population

characteristics. The initial population is constructed according to the initial data and constraints of the problem. The initial population will be updated throughout the algorithm iterations.

Step 2: Evaluate individuals A fitness function is used to evaluate the individuals of the populations in order to test their distance to the optimal solution. The fitness is carefully designed in order to make the EA process reach the final solution by maintaining the individuals whose fitness values are better.

Step 3: Mutation A mutation process is performed; to explore unseen pool of solutions. When a solution "mutates", a random gene is changed to another random value. Mutations happen with a very low frequency and, as in real life, are usually "destructive" for the individual (that is, its fitness value usually decreases). However, mutations are needed to create diversity and to inject new solutions into the population.

As seen in Figure 2.4, without mutation the solution pool could wrongly converge to a "local optimal solution". A successful mutation can create completely random solutions, leading to "unexplored" solutions that cannot be reached in other ways. [6]

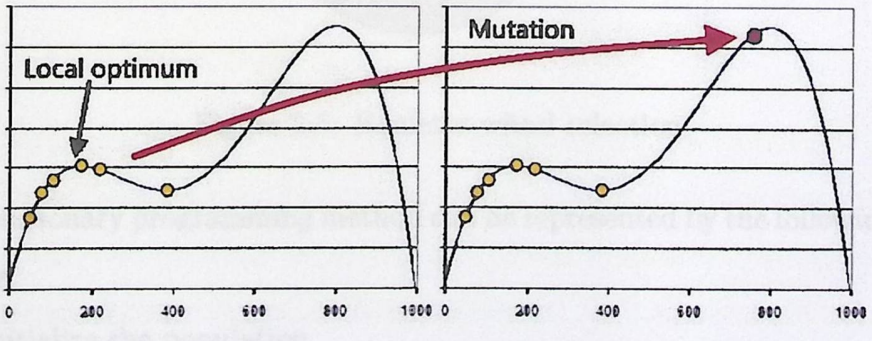


Figure 2.4: Mutation

Step 4 : Select members of new population The selection of the individuals of the next population depends on the calculated fitness values. Many

selection strategies can be implemented in the evolutionary program to determine which individuals to pass to the next generation such as Elitist selection, Scaling selection, Rank selection, Hierarchical selection, and Roulette-wheel selection. In our work we have used the roulette-wheel selection where the conceptualization is that of a wheel whose surface is subdivided into wedges representing the probabilities for each individual, look at Figure 2.5 For instance, one point on the edge is determined to be the zero point, and each arc around the circle corresponds to an area on the number line between zero and one. A random number is generated, between 0.0 and 1.0, and the individual whose wedge contains that number is chosen. In this way, individuals with greater fitness are more likely to be chosen. The selection algorithm can be repeated until the desired number of individuals has been selected. In roulette wheel selection, the probability of an individual being selected (the size of its slice of the wheel) is proportional to its fitness (indicated here by shading).

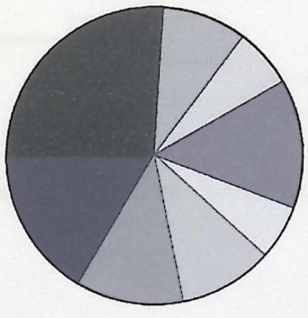


Figure 2.5: Roulette wheel selection

Evolutionary programming method can be represented by the following pseudo-code:

```
Initialize the population
Evaluate initial population
repeat
    Perform competitive selection
    Apply Mutation
```

Evaluate solutions in the population

until some convergence criteria is satisfied

The flowchart in Figure 2.6 show the EA.

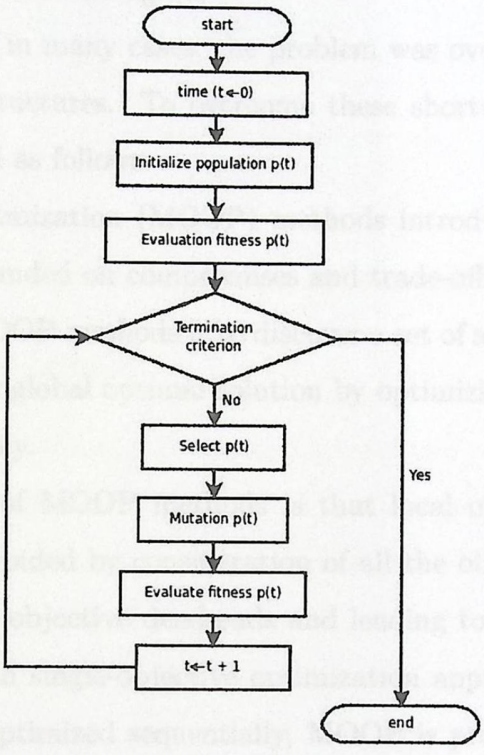


Figure 2.6: The flowchart of the evolutionary algorithm

2.2.2 Multi-objective

The first implementation of muti-objective optimization was in the mid-1980s, recently there has been a growing interest in evolutionary multi-objective optimization algorithms which focus on two important disciplines: evolutionary computation, and the area of multi-objective optimization. Most real problems can not be defined in single objective problems. Instead, they are defined with heterogeneous objectives which give us a better idea of how to solve them [1].

Though the academic class timetabling problem is being studied for more than four decades, a general solution technique for it, considering different aspects of its

variants, is yet to be formulated. Despite multiple criteria to be met simultaneously, the problem was generally tackled as single-objective optimization problem. Moreover, most of the earlier works were concentrated on school timetabling, and only a few on university class timetabling. [1]

On the other hand, in many cases, the problem was over-simplified by skipping many complex class-structures. To overcome these shortages the multi-objective approach was identified as follows.

Multi-objective optimization (MOOP) methods introduce a new approach for optimization that is founded on compromises and trade-offs among the various objectives. The aim of MOOP methods is to discover a set of satisfactory compromises and, through them, the global optimal solution by optimizing numerous dependent properties simultaneously.

The major benefit of MOOP methods is that local optima corresponding to one objective can be avoided by consideration of all the objectives simultaneously, thereby escaping single objective dead-ends and leading to a more efficient overall process. Compared with single-objective optimization approaches where each of a series of objectives is optimized sequentially, MOOP is proven to produce a more representative set of equivalent solutions faster and thereby allowing users to make informed decisions related to all objectives under consideration.

As seen in Figure 2.7 [13] A bi-objective (binary objective) problem is a simplified form of general multi-objective problem. Each point represents a solution to the bi-objective problem. The curved line represents the Pareto-front (where optimal solutions can be found), identified non-dominated solutions are labeled with a Pareto rank of zero, and Pareto rank of the other solutions refers to the number of solutions dominating it. Note that the problem requires minimization of both objectives.

Solutions which have number 1 is not Pareto optimal (optimal solution) as solutions which have number 2 has simultaneously smaller values for both objectives. There is no reason why solution 1 should be accepted rather than solution 2. Therefore the aim of MOOP is to obtain a representative set of non-dominated solutions.

variants, is yet to be formulated. Despite multiple criteria to be met simultaneously, the problem was generally tackled as single-objective optimization problem. Moreover, most of the earlier works were concentrated on school timetabling, and only a few on university class timetabling. [1]

On the other hand, in many cases, the problem was over-simplified by skipping many complex class-structures. To overcome these shortages the multi-objective approach was identified as follows.

Multi-objective optimization (MOOP) methods introduce a new approach for optimization that is founded on compromises and trade-offs among the various objectives. The aim of MOOP methods is to discover a set of satisfactory compromises and, through them, the global optimal solution by optimizing numerous dependent properties simultaneously.

The major benefit of MOOP methods is that local optima corresponding to one objective can be avoided by consideration of all the objectives simultaneously, thereby escaping single objective dead-ends and leading to a more efficient overall process. Compared with single-objective optimization approaches where each of a series of objectives is optimized sequentially, MOOP is proven to produce a more representative set of equivalent solutions faster and thereby allowing users to make informed decisions related to all objectives under consideration.

As seen in Figure 2.7 [13] A bi-objective (binary objective) problem is a simplified form of general multi-objective problem. Each point represents a solution to the bi-objective problem. The curved line represents the Pareto-front (where optimal solutions can be found), identified non-dominated solutions are labeled with a Pareto rank of zero, and Pareto rank of the other solutions refers to the number of solutions dominating it. Note that the problem requires minimization of both objectives.

Solutions which have number 1 is not Pareto optimal (optimal solution) as solutions which have number 2 has simultaneously smaller values for both objectives. There is no reason why solution 1 should be accepted rather than solution 2. Therefore the aim of MOOP is to obtain a representative set of non-dominated solutions.

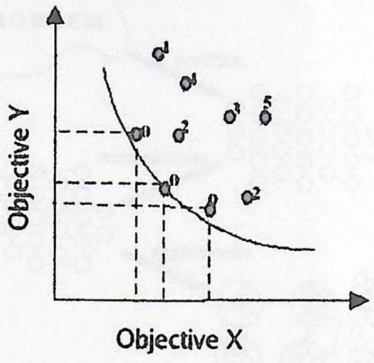


Figure 2.7: Multi-objective optimization demonstration

2.2.3 Parallel algorithm

"A parallel computer is a set of processors that are able to work cooperatively to solve a computational problem." That includes: the supercomputers (which have large number of processors), workstations connected by network, multiple-processor workstations, and embedded systems [9].

In the sequential programming, the instructions executed one by one as a series, on the other hand the parallel programming is to divide the program as a set of parts distributed into a cluster of machines (parallel computers) and executed in parallel (concurrently), that will reduce the execution time.

One of the most important issues in parallel computing is the load balancing, which means redistributing the tasks to get the best efficiency possible. The load balancing importance appears clearly in the heterogeneous environment systems, where every node has its own specifications.

As a software engineering issue, building a parallel program is divided into four stages which are partitioning, communication, agglomeration and mapping as shown in Figure 2.8 [9]. Each stage has a lot of details and we will discuss it to determine which techniques will be used.

Partitioning: The aim of partitioning, or decomposition, is to divide the main task of the program into a number of small tasks, there are two main approaches

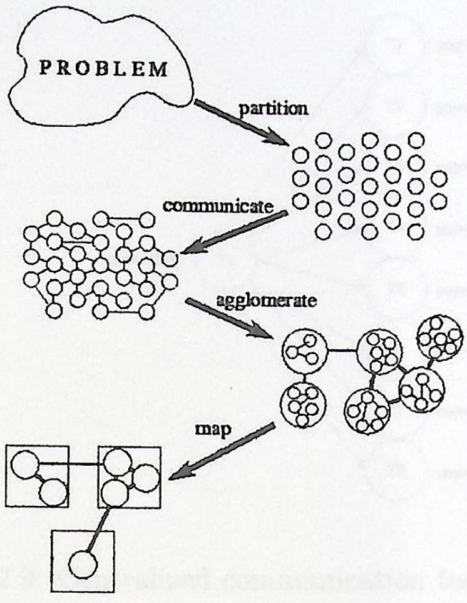


Figure 2.8: A design methodology for parallel programs

to partitioning the program. We can partition the computation, which known as Functional Decomposition and it divides the computation into disjoint tasks, or the data, which known as Domain Decomposition which involves the division of data into a set of small data size, then distribute the computation to be associated with the data. This yields many tasks, which needs to communicate with each other to exchange the data load and some information [9].

Communication: When we decompose the program into small tasks, some task (consumers) may need some data or information form other (producers), so a communications are needed. The communications may be categorized to be: local/ global, structured/ unstructured, static/ dynamic and synchronous/ asynchronous. In additoin to that two techniqes may be used to connect the node with each other, the first is global centralized communication (Figure 2.9), as in any centralized system this approach suffers from many problems, the most popular one is the single-point of failure, the second one called divide and conquer communication (Figure 2.10), which can avoid both links of the centralized approach. But the problem in the divide and conquer technique, is that the root can not communicate with all processes directly, which means, a long delay [9].

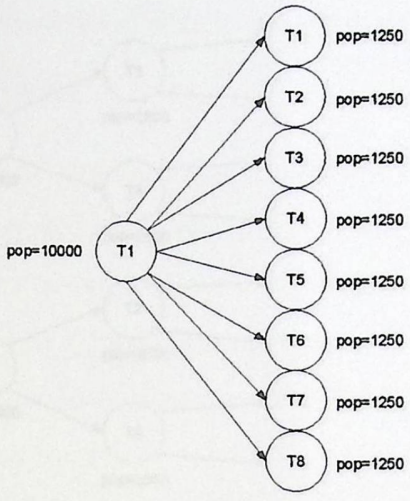


Figure 2.9: Centralized communication techniques

Agglomeration: The agglomeration phase is to combine the tasks identified by the partitioning phase, to provide a smaller number of tasks. Its goals are reducing the communication and computation costs, retaining flexibility, and reducing software engineering costs.

Typically, the computation is stopped when there is a message is being sent or received becouse the execution may depend on this message, therefore, we can improve the performance by sending less data or by using fewer messages. Another issue is to make a trade off between replicated computation for reduced communication requirements and/or execution time. An additional software engineer issue appear when there sequential algorithm exist, the parallelizing strategies may not make big changes. Another issue must be considered when the parallel algorithm executes as a part of a large system, then the data distributions utilized by other program components [9].

Mapping: The goal of mapping algorithms is to minimize the execution time, two strategies are used to achieve this goal, and we must make a trade-off between them:

1. The first is to place the tasks that are able to execute concurrently on different processors.

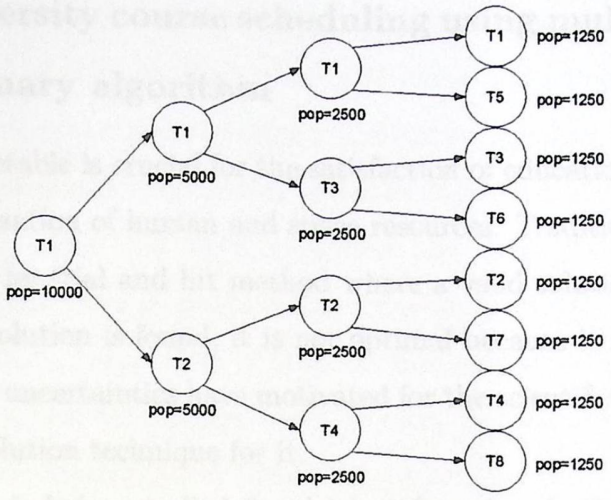


Figure 2.10: divide and conquer communication techniques

2. And the other is to place the tasks that communicate frequently on the same processor.

Three levels of mapping complexity, categorized by domain decomposition, the first if the tasks divide into fixed number of equal-sized tasks and structured local and global communication, in this case we map the tasks in a way that minimizes interprocessor communication. The second which is more complex algorithms if the tasks has variable size and/or unstructured communication patterns, in this case we may use a load balancing algorithm. Finally, The most complex problem happens when either the number of tasks or the amount of computation or communication per task changes dynamically during program execution, in this case we may use a dynamic load balancing algorithm [9].

2.3 Previous works

This section presents the related works done in the area of evolutionary algorithm multi-objective optimization and parallel programming.

2.3.1 University course scheduling using multi-objective evolutionary algorithm

An effective timetable is crucial for the satisfaction of educational requirements and the efficient utilization of human and space resources. Traditionally, the problem is solved manually by trial and hit method where a valid solution is not guaranteed. Even if a valid solution is found, it is not optimal because it may miss even better solutions. These uncertainties have motivated for the scientific to study and develop an automated solution technique for it.

The problem is being studied for the last four decades but a general solution technique for it is not yet formulated . The problem was first studied by Gotlieb et, al In their work, it was represented using class-teacher timetabling by considering that each lecture contained one group of students, one teacher, and any number of times which could be chosen freely [7].

The scheduling problem in schools is relatively simpler than UCS because of their simple class structures, therefore classical methods, such as linear programming approaches [7], could be used easily to solve the scheduling problem.

However, in the case of colleges and universities, which contain different types of complicated class-structures, the complexity of the problem is increased. As a result, classical methods can not manage to solve the problem specially in reaching multiple objective functions. These drawbacks of the classical methods have drawn the attention of the researchers towards the heuristic-based non-classical techniques [7], such as simulated annealing, tabu search and genetic algorithms [21].

"Once the class timetabling problem has been identified as an optimization problem, we need one or more objectives to optimize. However, an objective function in this problem is just an arbitrary measure of the quality of a solution (Abramson and Abela 1992). Hence, the choice of objectives varies from university to university. Most of the researchers treated the problem as a single-objective optimization problem, and took the minimization of total constraints violation as the only objective function (Abramson and Abela 1992; Blum et al. 2002; Lima et al. 2001;

Piola 1994). However, various choices of objectives, such as a compact timetable, minimum number of consecutive classes of teachers and so on, lead the scheduling of a class timetable to a multi-objective optimization problem. However, a very limited number of researchers considered multiple objectives in the problem (Paquete and Fonseca 2001; Silva et al. 2004). Carrasco and Pato (2001) used a bi-objective model to school timetabling problem for minimizing violation of soft constraints from two competitive perspective of teachers and classes. Filho and Lorena (2001) also tackled the school timetabling problem using bi-objective model where teachers and classes are clustered, and their conflicts are minimized. Desef et al. (2004) used another bi-objective model to German primary schools. Both of their objective functions are the minimization of weighted sum of idle time-slots and the extents to which the daily classes are finished, where higher priority is given to the earlier one in one objective function, and the later in the other objective function." [17]

2.3.2 Parallel evolutionary algorithms

A client-server architecture which implements a parallel hybrid genetic algorithm, two types of clients are used, one optimization-client (O-client) for optimization and the other Gene Expression Programming client (GEP-client) which used to implement the GEP methodology to solve the problem by using the parameters given from the O-clients based on fitness provided from the server. The GEP-client is used for the communication between the server and the clients according to the following cycle [16]:

1. An O-client sends a run parameters to the server. The server stores the parameters in a pool.
2. The server selects a run parameters from its pool and sends them to a GEP client; the client starts a GEP run.
3. A GEP-client finishes the run; statistics for that run are sent to the server.

Piola 1994). However, various choices of objectives, such as a compact timetable, minimum number of consecutive classes of teachers and so on, lead the scheduling of a class timetable to a multi-objective optimization problem. However, a very limited number of researchers considered multiple objectives in the problem (Paquete and Fonseca 2001; Silva et al. 2004). Carrasco and Pato (2001) used a bi-objective model to school timetabling problem for minimizing violation of soft constraints from two competitive perspective of teachers and classes. Filho and Lorena (2001) also tackled the school timetabling problem using bi-objective model where teachers and classes are clustered, and their conflicts are minimized. Desef et al. (2004) used another bi-objective model to German primary schools. Both of their objective functions are the minimization of weighted sum of idle time-slots and the extents to which the daily classes are finished, where higher priority is given to the earlier one in one objective function, and the later in the other objective function." [17]

2.3.2 Parallel evolutionary algorithms

A client-server architecture which implements a parallel hybrid genetic algorithm, two types of clients are used, one optimization-client (O-client) for optimization and the other Gene Expression Programming client (GEP-client) which used to implement the GEP methodology to solve the problem by using the parameters given from the O-clients based on fitness provided from the server. The GEP-client is used for the communication between the server and the clients according to the following cycle [16]:

1. An O-client sends a run parameters to the server. The server stores the parameters in a pool.
2. The server selects a run parameters from its pool and sends them to a GEP client; the client starts a GEP run.
3. A GEP-client finishes the run; statistics for that run are sent to the server.

4. The server sends the statistics back to the O-client.

5. An O-client uses meta-heuristics to tune the input parameters of the run based on the statistics (fitness of the parameters) sent by the server.

The parallelization of the GEP methodology in [16] was based on an island model with migration. The islands are connected according to a fully connected topology. In this migration model, each population selects an individual amongst the best individuals of the best individuals of the other populations, and this individual replaces its worst individual. This is repeated for each population.

In [8], the author make a comparison between Message Passing programming and Shared Memory. One of the MPI disadvantages is its special programming style, where it requires explicit communication between processes, on the contrary shared memory did not require any special programming style. On the other hand, MPI provides better performance than Shared Memory, except under some conditions, the performance gap will be closer.

They do three experiments in [8], a sequential EA, MPI Parallel Evolutionary Algorithm (PEA), and shared memory PEA. They implement PEA in the master/slave architecture because it's easy for implementation and its behavior is easy to be understood as they mentioned.

The most intuitive implementation of PEA is to divide population into several chunks of equal size and distribute each of them to every processor who calculates the fitness of each individual in the assigned chunk. This model of implementation is called a single-population master-salve evolutionary algorithm where a processor that keeps the population is a master and all the rest are slaves. The master processor performs selection and crossover operations for the entire population. Also, there are fine-grained and coarse-grained models of PEA. Fine-grained PEA contains a single population but selection and crossover are limited to small overlapping neighborhoods. This kind of PEA works very efficiently with massively parallel computers. Coarse-grained PEA, sometimes called an Island-Model, is more sophisticated than the previous two models because it maintains multiple sub-populations

and also allows migration of individuals among them. As each processor performs selection and crossover operations independently on its own population and only exchanges individuals occasionally, the communication overheads required in this model are much less than in the others, making it the most popular method for implementing PEA [8].

In [5], the authors implement a PEA to solve the Traveling Salesman Problem (TSP). Two parallelization models are proposed in this paper, Independent model, were each process execute the algorithm on a suppopulation with variouse parameters, at the end, the best solutions collected to find the best of the bests. The advantage of this model is the diversification of the solutions space search process.

The second model works in the same way as a sequential algorithm with a big population composed of subpopulations of each process. Processes are connected with independent evolutionary algorithms and different subpopulations. The same parameters are used in every process. In every iteration the average number of permutations (for all subpopulations) in which there is an element a in the position is computed [5].

As mentioned in [5], the authors implement the PEA in c++ language with mpi library (MPICH) and it tested on the cluster of 13 computer with the same archeticture. Both seqential and parallel algorithm consume the same calculation time.

Chapter 3

Problem modeling and methodology

3.1 Overview

In this chapter we will present the problem description and its modeling, then we will represent the solution that we applied to solve the problem, after that we will describe the implementation of the evolutionary steps in our project, demonstrate the timetabling system, class diagram, finally we will view the flowcharts of every function of the proposed evolutionary algorithm.

3.2 Problem description and modeling

In this project we will start working on the timetable design problem for one of the four collages of the university which is the College of Administrative Sciences and Informatics, then the solution will be generalized on the rest of collages. In this college there are four academic 4-years programs, which are: Information Technology (IT), Information System (IS), Graphics and Multimedia (GM), and Business Administration (BA). So, courses are offered at the beginning of each semester for 4 student groups in each academic program, so we have 16 total student groups (4

levels * 4 programs). On the other hand, we have five work days a week (Sunday to Thursday). On Sunday, Tuesday, and Thursday there are nine 60- minutes time slots a day, and on Monday and Wednesday there are six 90-minutes time slots a day.

To handle the problem correctly, we have defined the problem to be six different sets: student groups, instructors, course sections, class rooms, timeslots (lectures), and a set of constrains. Then the problem is formulated as following: {S, I, C, R, L, O}, where: S = $\{s_1, s_2, ..., s_i\}$ is the set of student groups, each element in S is a vector (m, y, std_slot) where m: is the major (IT, IS, GM, or BA), y: is the group number which will be represented by academic year of that group, this vector determines a group of students of the same major and academic year. So, for the considered collage, m ranges from 1 to 4, and y ranges from 1 to 4, and std_slot is an array of timeslots that indicates when the student groups are available. I = $\{i_1, i_2, ..., i_j\}$ is the set of instructors in the college at the current semester, each element in I is a vector (inst_no, inst_slot) where inst_no: is the number of a specific instructor, inst_slot: is an array indicating to timeslots when the instructor can give a lecture. C = $\{c_1, c_2, ..., c_n\}$ is the set of course sections offered for the student in the current semester, each element in C is a vector (s_i, co_no, section, cap, t_j, th, ph, ph_type, ph_inst) where s_i: is the group of students, which this course is offered for, co_no: is the course number, section: is the section number of this course, cap: is the maximum capacity of the section, t_j: is the instructor of this section, th: is the number of theoretical hours of this course, ph: is the number of practical hours of this course, ph_type: is the type of the practical hour it is set to 0:if the course has no practical hour,1: if this hour is lab, set to 2 if this hour is multimedia lab, set to 3 if this hour is photo lab, and set to 4 to indicate that this hour is workshop, ph_inst: the period that the teacher must be in the lab it is set to 0: no hour in the lab, 1: half of the lab period, 2: all the lab period.

R = $\{r_1, r_2, ... , r_m\}$ is the set of classrooms, each element in R is a vector (cap, d, t) where cap: is the maximum capacity of this room (number of students can be

assigned to this classroom), d: is a flag set to 1 if this room has data show, t: is a flag used to determine the type of this classroom, set to 0 to indicate that the classroom is a room, set to 1 to indicate that the classroom is PC lab, set to 2 to indicate that the classroom is a multimedia lab, set to 3 to indicate that the classroom is photo lab, and set to 4 to indicate that the classroom is workshop. L= $\{l_1, l_2, ..., l_k\}$ is the set of time slots of the week, each element in L is a vector (n, d) where n: is the time slot number, and ranges from 1 to 9 for days 1, 3, and 5, and ranges from 1 to 6 for days 2 and 4, and d is the day ranges from 1 to 5. Thus, L = { (1,1), (2,1), ..., (9,1), (1,2), (2,2), ..., (6,2), ..., (8,5), (9,5)} where (1,1) means: the first lecture in the first working day (Sunday), and so on. Each lecture (theoretical hour) on 1, 3, and 5 is 60 minutes, so the workable hour is 3 timeslots (3 hours) in these days. On 2 and 4 each lecture (theoretical hour) is 90 minutes, so the workable hour is 2 timeslots (3 hours) in these days. For example if the course has 3 theoretical hours then it takes 3 timeslots in days 1, 3, and 5 or 2 timeslots in days 2 and 4. Except on 1 and 5 the fifth lecture is 2 timeslots because on 3 there is no fifth lecture.

O = $\{o_1, o_2, ..., o_q\}$ is the set of constraints, where o_q: is the penalty weight (cost) of this constraint. Hard constraints will be assigned infinity cost, while soft constraints will assigned some constant cost to indicate the weight of violating that each one. The sets described above are used to define the problem; on the other hand the set of solutions of such problem is very large set. Each solution will be evaluated using an indicator that determines how much this solution is good (fitness). Our methodology will use an Evolutionary algorithm to search for some optimal solution.

As mentioned in , there are four stages to distribute the algorithms, here we will discuss the techniques which used at each stage.

In this project we use the domain decomposition, where the population will be divided into subpopulations which then is distributed among various processes, each of these processes will produce a set of solutions, then a main process will choose the optimal solution form the results of each one. In this case we find the solution form a large population, which mean that there is a chance for better solutions,

and a reduction execution time needed because each machine has small number of individuals. [7]

In this work, a master-worker model will be used where many worker processes should execute under the control of one master or root process, each process should communicate to the root process, each process will take the parameters from the root, make the processing and then return the results back to the root, which will make additional processing to get the optimal solution. Because the root should communicate with all processes, we will use global centralized communication technique.

In our case we use a dynamic load balancing algorithm because at the execution time we need to change the population size in may cases.

3.3 Solution definition

At this section we will describe our solution for the UCS problem.

3.3.1 Solution definition

G= $\{g_1, g_2, ..., g_w\}$ which represented in Figure 3.1 is the set of genes which constitute the individual (chromosome), each gene is a vector (course, room_no, ph_tslot, lab_no, th_tslot), look to Figure 3.2. where: course is the number of course section, room_no is the number of room assigned to this course section during theoretical hours, th_tslot is timeslot assigned to the course section for theoretical hours, lab_no is the number of lab assigned to the course during practical hours and ph_tslot is the timeslot (lab) assigned to the course during practical hours.

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

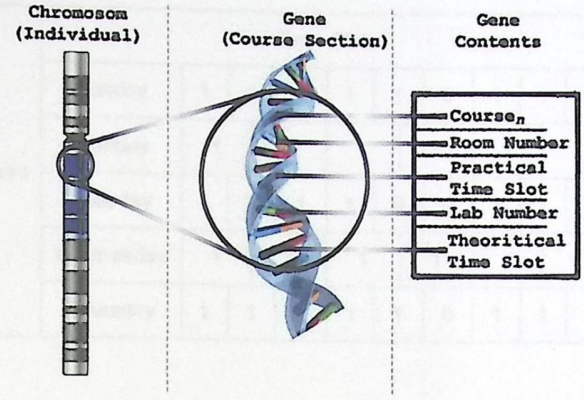


Figure 3.1: Chromosome and Gene representation

G_1	G_2	.....	G_i	....	G_w
$course^1$	$course^2$		$Course^i$		$Course^w$
room_no	room_no		room_no		room_no
ph_tslot	ph_tslot	.....	ph_tslot	.....	ph_tslot
lab_no	lab_no		lab_no		lab_no
th_tslot	th_tslot		th_tslot		th_tslot

Figure 3.2: The individual

for optimizing the solution considering the hard and soft constraints for the students and only the soft constraints for the lecturers. The individual is constructed in a way that every gene has different number of options when choosing timeslot and classroom; such that, course section in gene G1 has a maximum number options available when randomly selecting classrooms and timeslots, course section in gene G2 has 1 option less than those was available for gene G1, and the last one, gene Gw has the least number of option.

The instance chosen for our application has a maximum of 14 classrooms and 7 labs available. On the other hand, along the week, we have 41 theoretical classes (1 hour each) and 13 practical classes or labs (3 hours each). That means we have a maximum capacity of (41*14) = 574 theoretical classes, and (13*7)= 91 practical classes along the week. As shown in Figure 3.3, where 1 means time slot is available, and 0 means time slot is not available.

Constraints:

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

Figure 3.3: The timeslots

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

Hard constraints are:

- A teacher must not have more than one class at any given time slot. (Violation cost = infinity).
- At any given time slot no student group can have more than one class. (Violation cost = infinity).
- An instructor can be assigned classes only in his available time, like part timers. (Violation cost = infinity).
- A room must not assigned more than one class at any given time slot. (Violation cost = infinity).
- The number of students in any lecture should be less than or equal the maximum capacity of the classroom. (The cost increases as the number of students above the capacity of classroom increases, each student above the capacity increases the cost by a constant until the number of students above capacity is greater than a given small threshold, then the cost of this constraint will equal to infinity).

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

- Students should not have many classes consecutively without a break.
- Instructors should not have long free time between lectures.
- Students should not have long free time between lectures.
- Instructors should not have very late classes daily.
- A group of students should not have consecutive classes at different large distance locations.

3.3.2 Parallelization

There are two main reasons for paralleling an evolutionary algorithm: the first, is to minimize the execution time by distributing the computational effort. And the second, is to get benefit from a parallel setting from the algorithmic point of view, in analogy with the natural parallel evolution of spatially distributed populations [20].

In [12], they make definitions of the GA issues in parallel formulations.

Sequential run time T_s : is the time taken to solve a problem on a single processing element.

Parallel run time T_p : is the time taken to solve a particular problem instance on an ensemble of P processing elements [12].

Speedup S: it is the gain in computation speed achieved by using P processing elements with respect to a single processing element $S = T_s / T_p$ [12].

Efficiency E: It is the ration of the speedup to the number of processing elements used ($E = S/P$) [12].

Search overhead: it comes form communication overhead, idle time and contention for shared data structures [12].

Search overhead factor: it is the ratio of the work done by the parallel formulation to that done by the sequential formulation Wp /W where W the amount of work done

by a single processor and Wp the total amount of work done by P processors [12].

In the evolutionary parallel algorithm, the two most thing consume the time are the computation and the communication, and these have inverse relationship, so if we increases the communication, that will lead to reduce the computation and vice-verse [12].

Four classes of machines can be used with PGE, Single Instruction, Single Data (SISD), Single Instruction, Multiple Data (SIMD), Multiple Instruction Single Data (MISD) and Multiple Instruction, Multiple Data (MIMD). In the SISD corresponds to the classical mono-processor machine such as PCs [20], in MISD each processor may execute its own instructions with shared entry, in MIMD each processor may execute its own instructions with various entry [12] and in the SIMD (we will use this class in our parallelization) the same instructions will implemented in all processors with various entry [12], in other words, the same instructions will be broadcast to all processors, then each processors will execute the instructions in different data (Figure 3.4). The problem in the SIMD class, some processsors may be idle and waiting the other, if the problem domain is spatially or temporally irregular or if we work in a heterogeneous environment [20], so it will be necessary to balance the load.

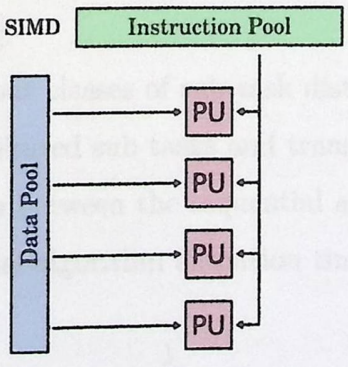


Figure 3.4: Single Instrucion Multiple Data

Two architectural feature related to parallel computation, message passing computer, and shared address space computer. "Shared address space parallel computers

have global memory that can be directly addressed by all processors. In message passing computers, each processor has its own local memory, and processors can communicate only by exchanging messages over the communication networks" (Figure 3.5) [12].

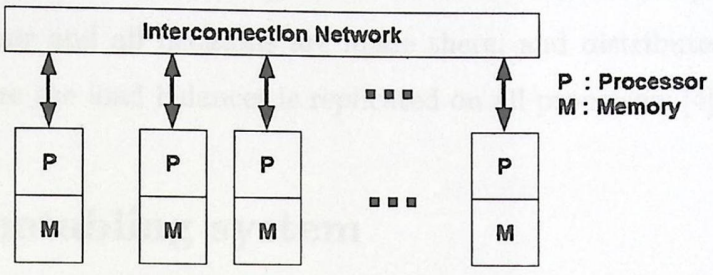


Figure 3.5: MPI architecture

Two schemes to generate distributed sub tasks, the first is *sender initiated sub task generation scheme*, where the generation of sub tasks is independent of the work requests from the idle processors, and the second is *receiver initiated sub task generation*, generation happen just when an idle processor requests for work [12].

Two ways to deliver the sub tasks into processors, either *Receiver initiated work transfer* what is on demand (when the processor idle), or *Sender initiated work transfer* without demand [12].

Note that we can make four classes of sub task distribution schemes from combination the generation distributed sub tasks and transformations way.

Amdahl's law: The ration between the sequential algorithm execution time (on one processor) and the parallel algorithm execution time (in many processors) [9].

Amdahl's law

$$\frac{1}{(1-P)+\frac{P}{N}}$$

where P is the proportion code that can be parallelize, and N is the number of processors.

Load balancing: "Load balancing means to distribute the workload of a parallel

application among the processors in the platform at hand according to their relative performance, in order to minimize the execution time of the program" [4].

Dynamic load balancing algorithms: it use state information to make decisions during program execution. Dynamic load balancing classes into two main classes: centralized dynamic load balancing, where the load balancer is implemented on one master processor and all decisions are made there, and distributed dynamic load balancing where the load balancer is replicated on all processors [4].

3.4 Timetabling system

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

In our work we have solved the problem under the object oriented principles using C++ language, so we have designed all the sets that describe the problem into classes which we will demonstrate each one as follow:

Element in S is a vector (m, y, std_slot) where m: is the major (IT, IS, G, or M), y: is the group number which will be represented by academic year of that group, this vector determines a group of students of the same major and academic year. So, for the considered collage, m ranges from 1 to 4, and y ranges from 1 to 4, and std_slot is an array of timeslots that indicates when the student groups are available.

I $=\{i_1, i_2, ..., i_j\}$ is the set of instructors in the college at the current semester, each element in I is a vector (inst_no, inst_slot) where inst_no: is the number of a specific instructor, inst_slot: is an array indicating to timeslots when the instructor can give a lecture.

1. Each object in the rooms class (S) contains the attributes shown in table 3.1.

2. Each object in the students groups class (S) contains the attributes shown in table 3.2.

application among the processors in the platform at hand according to their relative performance, in order to minimize the execution time of the program" [4].

Dynamic load balancing algorithms: it use state information to make decisions during program execution. Dynamic load balancing classes into two main classes: centralized dynamic load balancing, where the load balancer is implemented on one master processor and all decisions are made there, and distributed dynamic load balancing where the load balancer is replicated on all processors [4].

3.4 Timetabling system

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

In our work we have solved the problem under the object oriented principles using C++ language, so we have designed all the sets that describe the problem into classes which we will demonstrate each one as follow:

Element in S is a vector (m, y, std_slot) where m: is the major (IT, IS, G, or M), y: is the group number which will be represented by academic year of that group, this vector determines a group of students of the same major and academic year. So, for the considered collage, m ranges from 1 to 4, and y ranges from 1 to 4, and std_slot is an array of timeslots that indicates when the student groups are available.

I $=\{i_1, i_2, ..., i_j\}$ is the set of instructors in the college at the current semester, each element in I is a vector (inst_no, inst_slot) where inst_no: is the number of a specific instructor, inst_slot: is an array indicating to timeslots when the instructor can give a lecture.

1. Each object in the rooms class (S) contains the attributes shown in table 3.1.

2. Each object in the students groups class (S) contains the attributes shown in table 3.2.

Table 3.1: The class of rooms (rooms)

Name	Data type	Description
capacity	int	room capacity - number of students may set in this room
dataShow	short	is the room has a data show? 1 for yes and 0 for no
type	short	type of the room, 0: room, 1: PC lab, 2: multimedia lab, 3: photo lab, 4: Workshop
ts[R][C]	int	room time slot
roomId	int	room number

Table 3.2: The class of student groups (StdGroups)

Name	Data type	Description
major	int	A flag that set to 0:IT, 1:IS, 2:G, 3:M.
stdGroupId	short	student group number
year	int	2000, 2001, 2002, 2003, 2004,.....
ts[R][C]	int	The time slots for the student group.
softCost	int	student group's soft cost

3. Each object in the instructors class (I) contains the attributes shown in table 3.3.

Table 3.3: The class of instructors (instructors)

Name	Data type	Description
instId	int	The number of this instructor.
ts[R][C]	int	The time slots for the Instructor.
softCost	int	instructor soft cost

4. Each object in the courses class (C) contains the attributes shown in table 3.4.

5. Each object in the Genes class (G) contains the attributes shown in table 3.5.

Table 3.4: The class of courses (coursesSections)

Name	Data type	Description
CourseId	short	course number
section	Int	section number
capacity	short	The capacity of the course section
th	Int	Number of theoretical class hours for this course section
ph	int	Number of practical hours for this course section
phType	short	0: no practical hours, 1:PC lab, 2: multimedia lab, 3:photo lab, 4: workshop
phInst	short	The period teacher exist in this lab, 0: 0 hours, 1: 1.5hours, 2: 3 hours
stdGroup	int	The student group of this course section
instructor	int	The instructor of this section

Table 3.5: The class of genes (genes)

Name	Data type	Description
cost	long	The penalty of this gene
thTime	int	Theoretical time
phTime	int	Practical time
sCost	int	The soft cost of this gene
roomId	int	room number from rooms class
labId	int	lab room number from rooms class
courseId	int	course number form courseSections class

3.5 Class diagram

In this section, the class diagram is viewed in Figures 3.6, 3.7, 3.8 and 3.9. At this diagram we present the data members and the methods in this project. Note that there are association relation between classes, in addition to that we take the encapsulation benefit from the object oriented modeling.

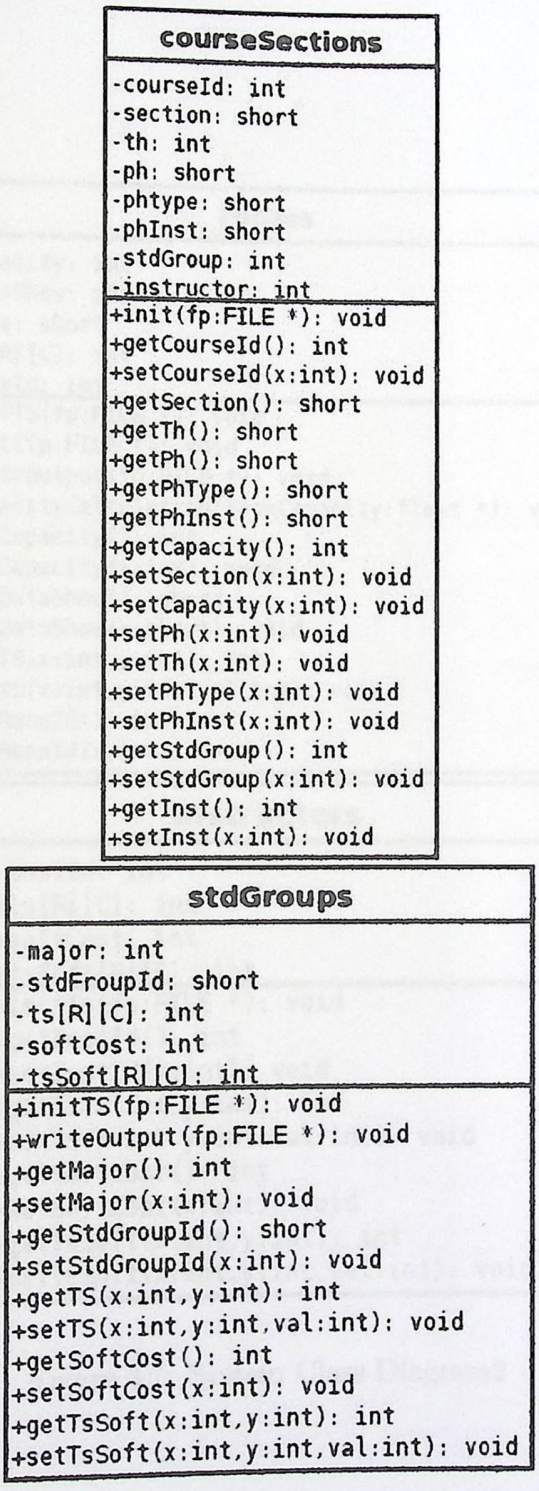


Figure 3.6: System Class Diagram1

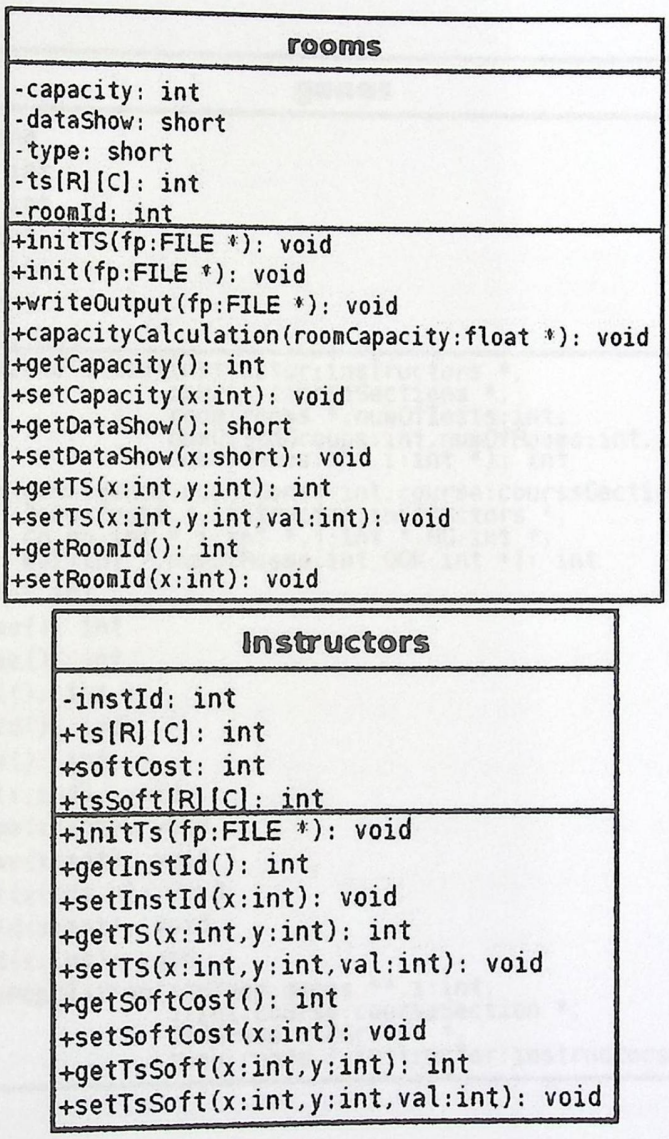


Figure 3.7: System Class Diagram2

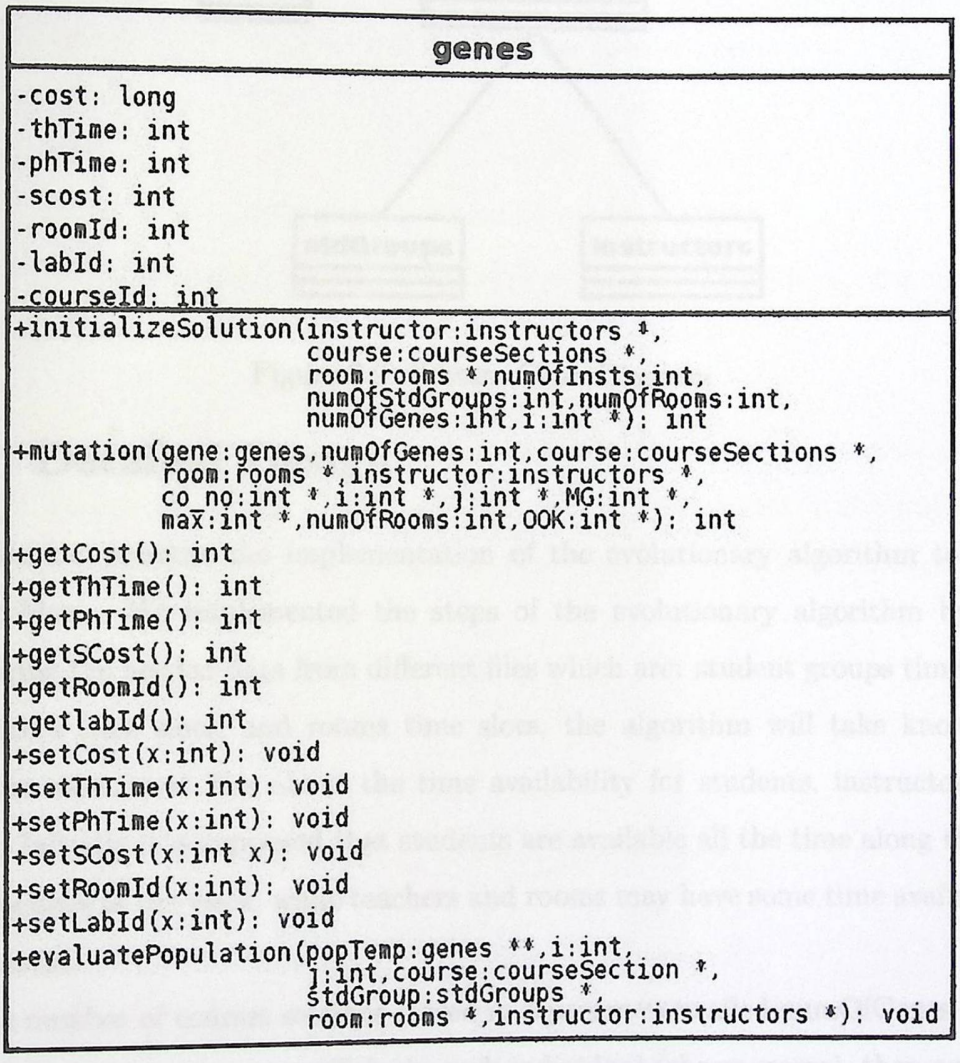


Figure 3.8: System Class Diagram3

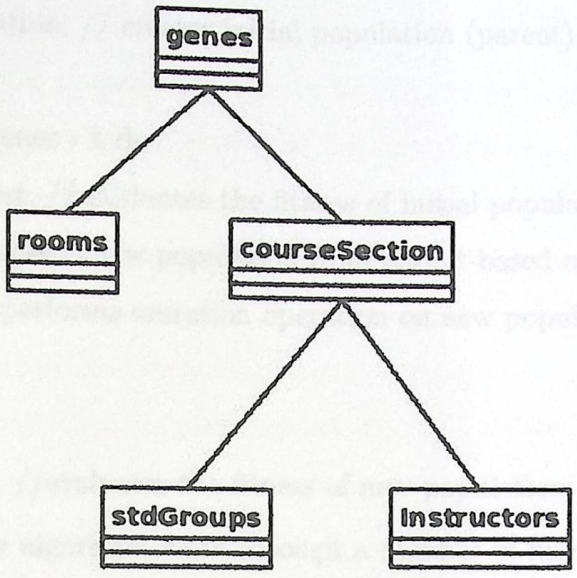


Figure 3.9: System Class Diagram

3.6 Detailed Design

This section presents the implementation of the evolutionary algorithm to solve the problem. We implemented the steps of the evolutionary algorithm by first initializing the needed data from different files which are: student groups time slots, instructors time slots, and rooms time slots, the algorithm will take knowledge from the mentioned files about the time availability for students, instructors and rooms. Initially it is supposed that students are available all the time along the five working days of the week, while teachers and rooms may have some time availability constraints.

The number of courses sections stored in a parameter called numOfGenes which determines how many genes will be in each individual (chromosome), then we built a dynamic array for the classes rooms, students groups, instructors, and courses sections. The number of individuals on each population is defined at the begining of the program called popSize, also we define a variable called numGener which determines the total iterations of the evolutionary algorithm. After that we built the evolutionary algorithm and its functions as shown in this algorithm:

Initialize_Population; // creates initial population (parent).

$t \leftarrow 0$;

while t ¡ numGener - 1 **do**

Population_cost; // evaluates the fitness of initial population.

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

Mutation; // performs mutation operation on new population.

$t \leftarrow t + 1$;

end while

Population_cost; //evaluates the fitness of new population.

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

First: Initialize_Population function creates the initial population of individuals based on the size of the population (popSize), and then we depend on this population in going through the algorithm iterations. Flowchart in Figure 3.10 explains.

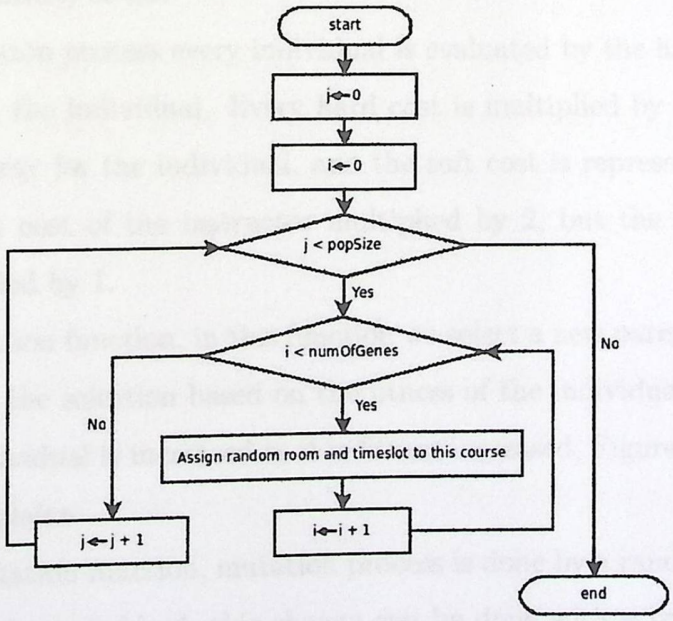


Figure 3.10: The flowchart of Initialize_Population function

Second: Evaluate_Population function evaluates the fitness of the population

as follow:

After we have built the population we evaluate the fitness of each individual in the population. Population cost is measured as the sum of the cost for each gene in the individual. The cost is determined by scanning the genes of the individual orderly about the violated constraints (hard and soft ones) by every gene; for example a hard constraint violation may happen if, in some gene, a course section is assigned in an occupied time slot for the teacher, room, or a student group. On the other hand a soft constraint violation may happen if a course section is a assigned in a time slot that makes long time wait, or too late, for the teacher of the course or the students. Figure 4.5 view the flow chart.

The fitness is calculated for the individual x by the simple function

$$f(x) = \frac{1}{cost(x) + 1}$$

where the cost(x) is the sum of the hard cost and soft cost of this gene. The value of this function is important in determining whether this individual will pass to next step (new population) or not.

In the evaluation process every individual is evaluated by the hard and soft cost of every gene on the individual. Every hard cost is multiplied by ∞ and added to the hard cost array for the individual, and the soft cost is represented through its priority, the soft cost of the instructor multiplied by 2, but the soft cost for the students multiplied by 1.

Third: Selection function, in this function we select a new parent population for next generation, the selection based on the fitness of the individual the probability to select this individual is increased as it is fitness increased, Figure 3.12 which view the flowchart explains.

Fourth: Mutation function, mutation process is done by a random modification on the genes in the individual; this change can be done with a probability. Every individual in population is ordered according to the genes hard and soft cost, where genes of zero cost in both are put at the beginning of the individual then the mutation starts from the first gene whose hard or soft cost is greater than zero. This operation

is done based on the probability (Mu) as shown in Figure 3.13.

In Figure 3.14 we view abstractly the system takes, two main blocks included, the evolutionary algorithm block and load balancing and communication block thats in additions to get the input data and return the output data blocks. Two main cases in this distributed algorithm, the first one if the processor is the ROOT, it will get the data, do the evolutionary algorithm as other processors, optimize best solution for the returned results, and write the output.

The other case if the processor is normal one (not ROOT/ client machine), it will get the parameters form the ROOT then do the evolutionary algorithm and return the result to the ROOT.

To balance the load between the processors, we use the *Exploitation of the Fastest Processor* (EFP) which shown in Figure 3.15 [4]. In this project, the most important part is the load balancing, where the system must use the cluster with the most utilization possible to get the results as faster as possible [4].

This algorithm named each processor, where the fastest processor is the ROOT, it contain the load balancing producer, in addition to do the optimization algorithm, this processor named by P_0. The other processors named P_1,P_2,...,P_{P-1} where P is the number of processors, and P_1 is the slowest processor and P_{P-1} is the fastest [4].

In this algorithm, the problem divided into G takes, where

$$G(P) = \lceil n * P * rand() \rceil * P * \frac{P-1}{2}$$

, $\lceil x \rceil$ is the least integer larger than or equals to x, n is a positive integer between 1 and 3, and $rand()$ is a function that returns a real number between 0 and 1. The tasks located into a queue to be ready to distribute [4].

The ROOT processor distribute the tasks to the other processors after that it will execute some tasks while it wait the other to return the results. If one of the processors do, that will interrupt the ROOT process to receives the results and send others tasks [4].

When the queue become empty, the ROOT will optimize the best solution, and then finish the algorithm by return the result to the user [4].

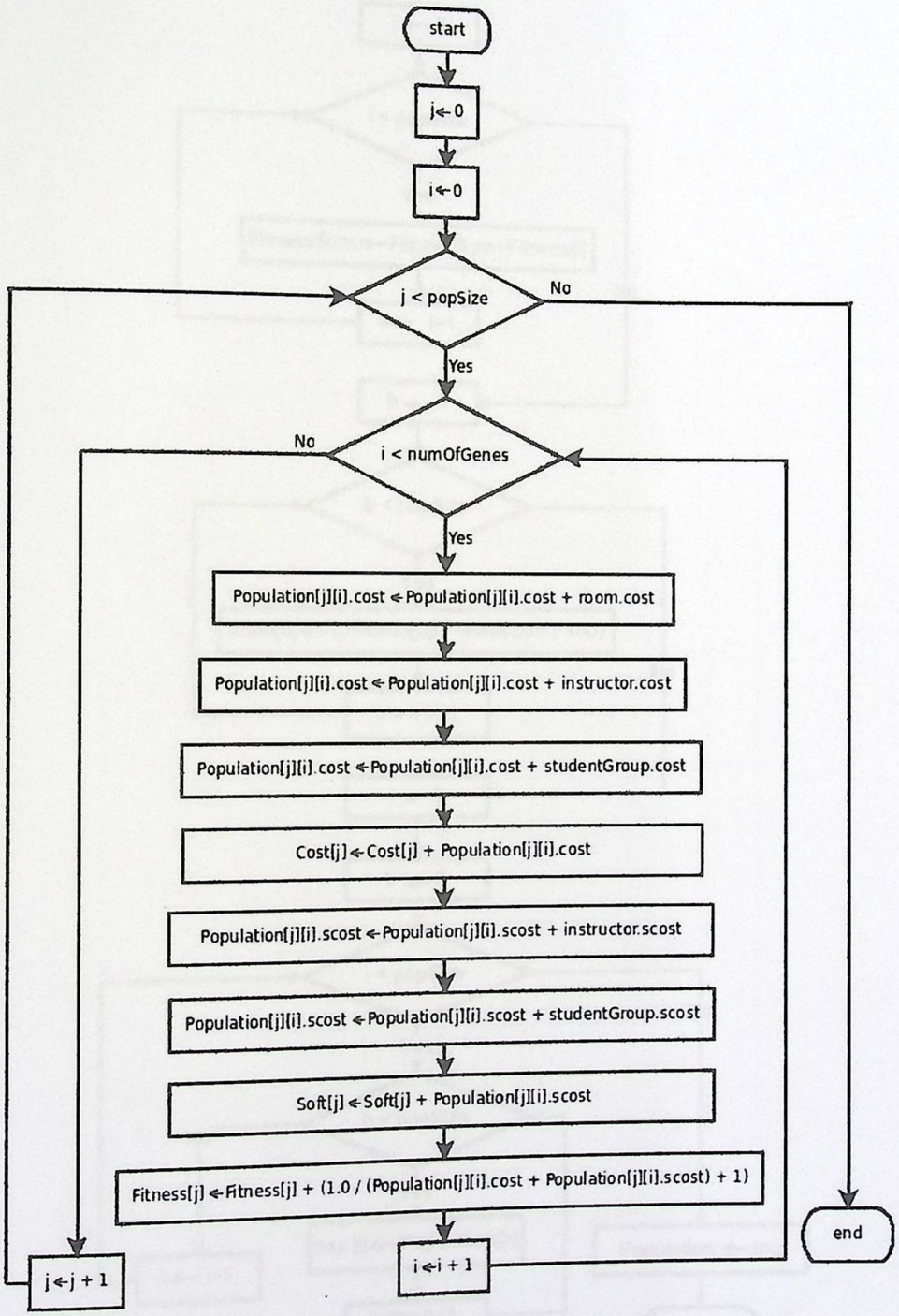


Figure 3.11: The flowchart of Evaluate_Population function

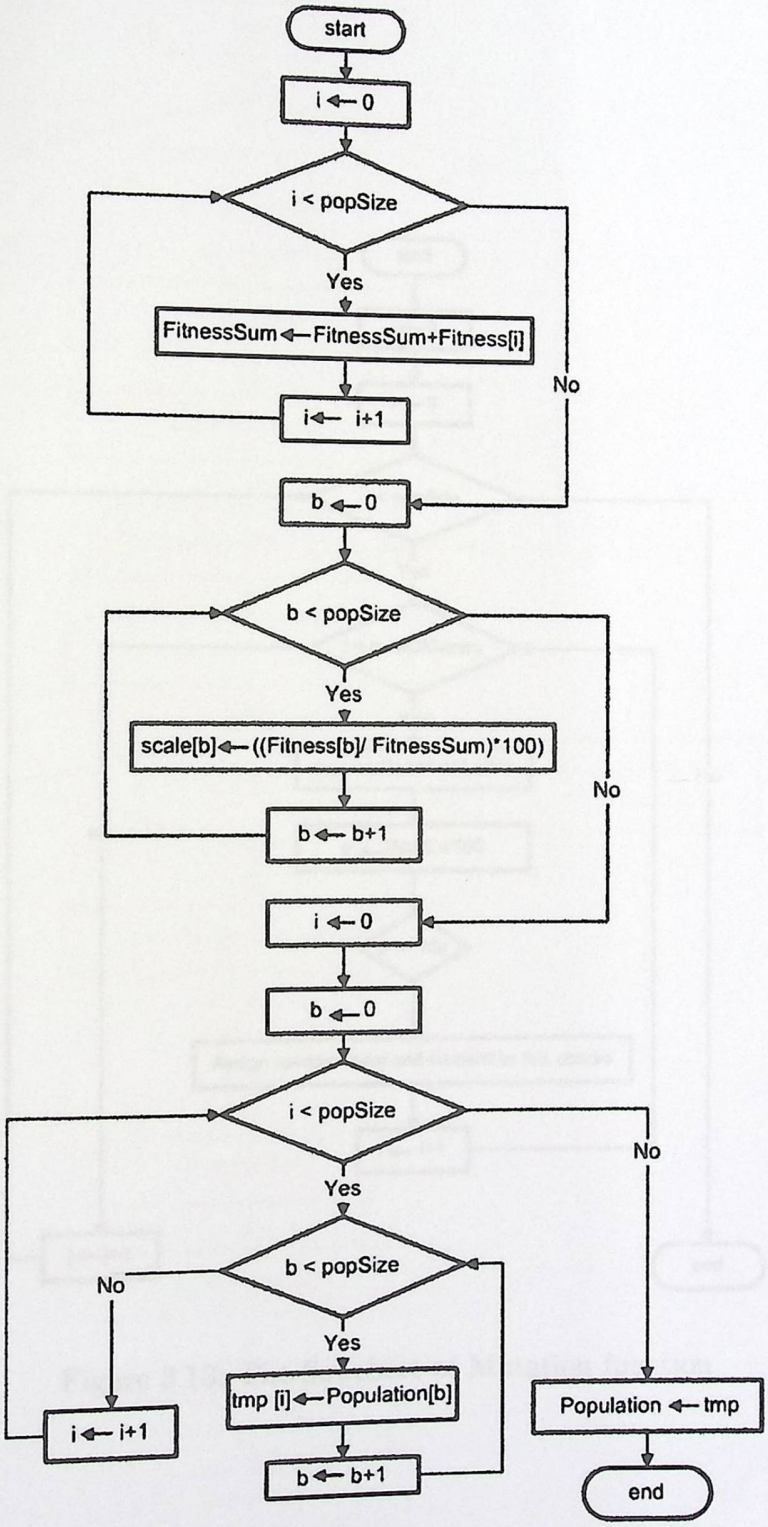


Figure 3.12: The flowchart of selection function

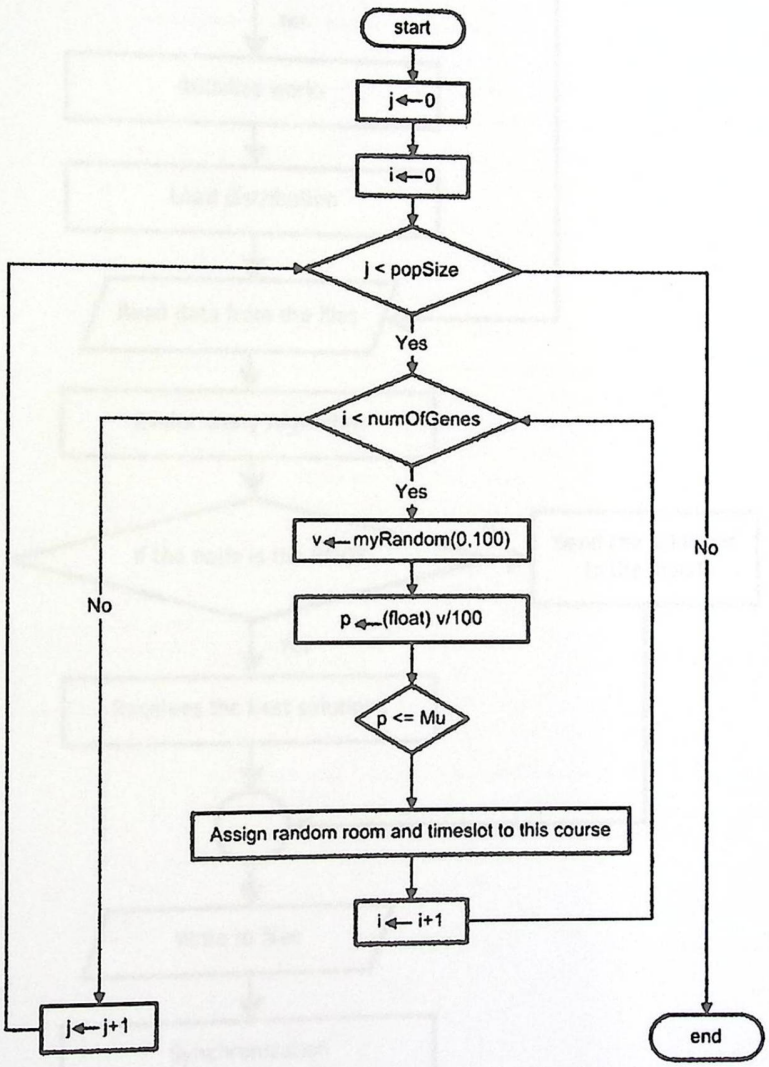


Figure 3.13: The flowchart of Mutation function

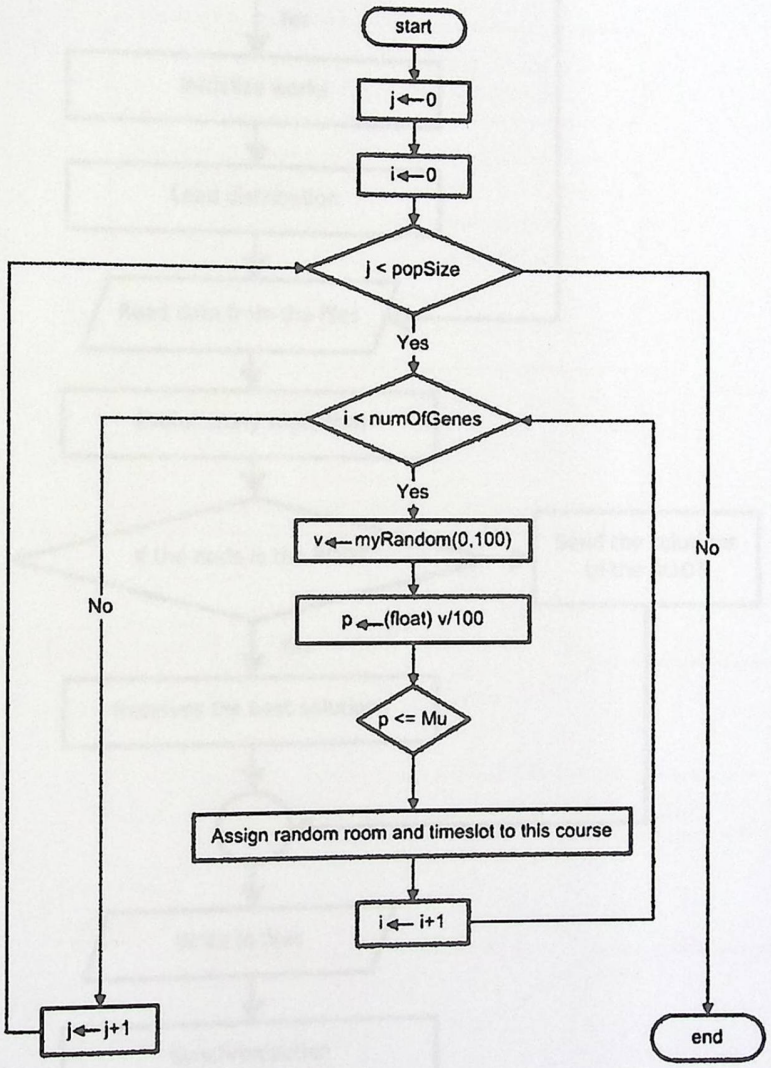


Figure 3.13: The flowchart of Mutation function

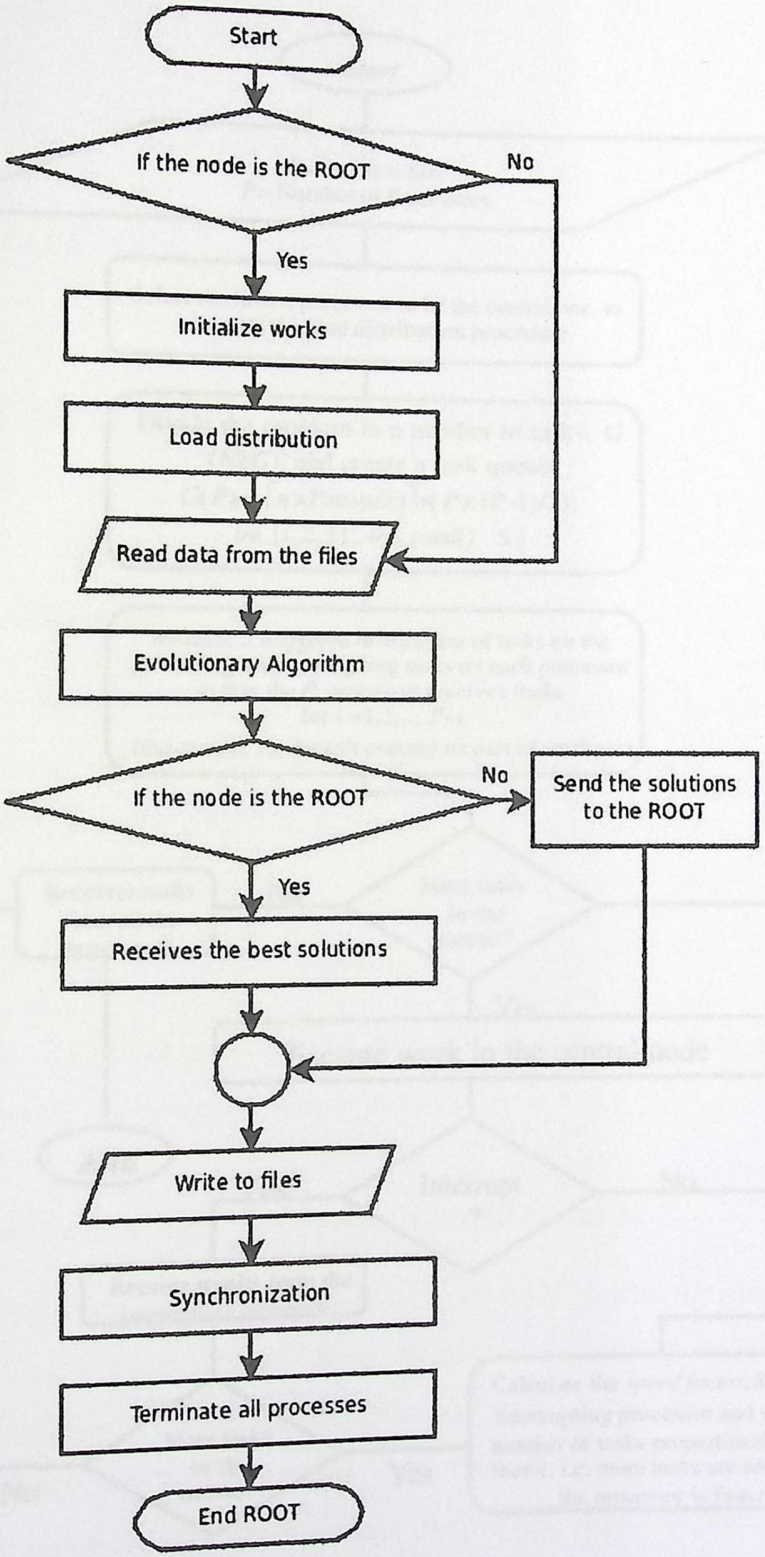


Figure 3.14: Flow chart for the proposed algorithm

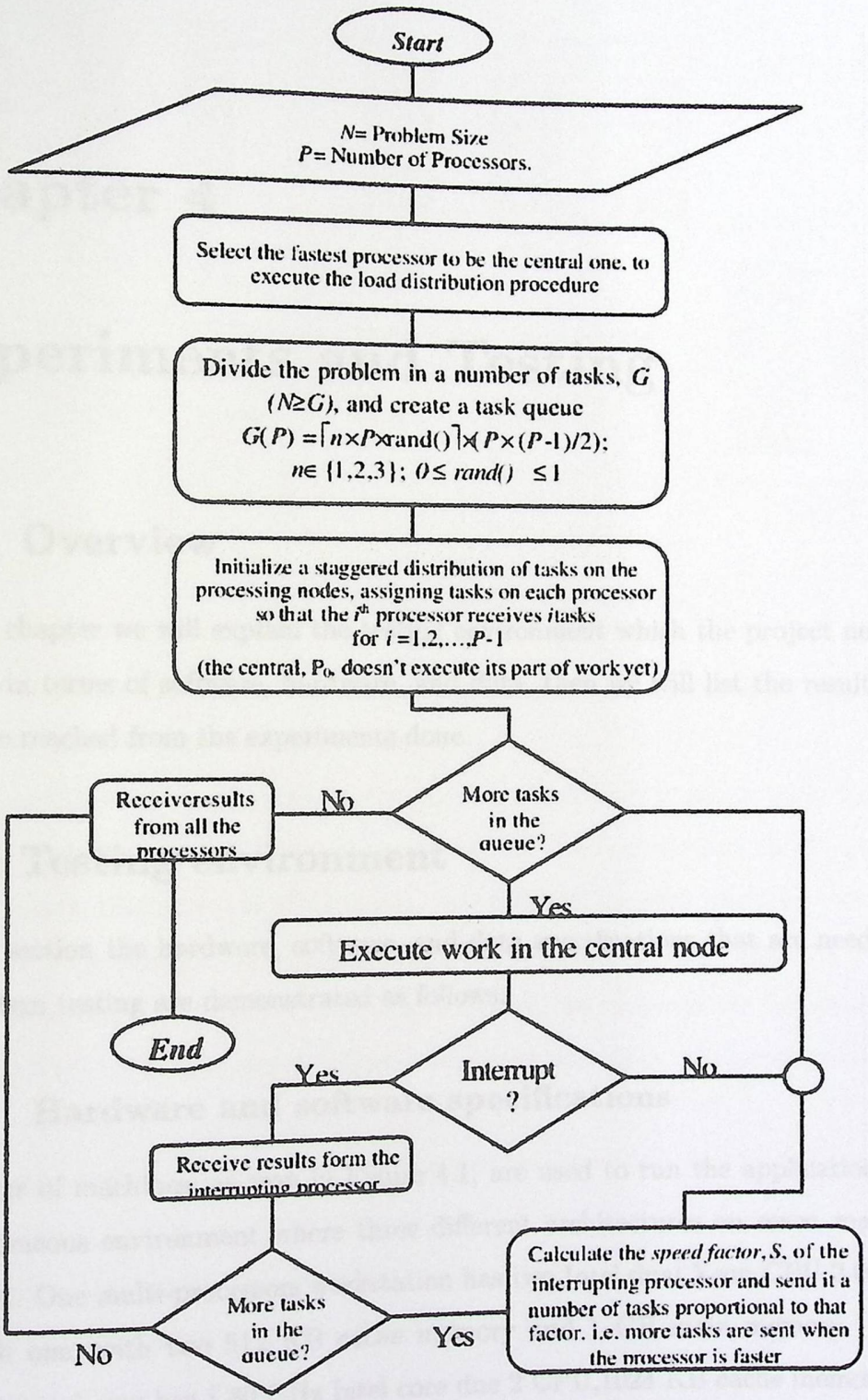


Figure 3.15: Flow chart for parallization technique with load balancing

Chapter 4

Experiments and Testing

4.1 Overview

In this chapter we will explain the testing environment which the project needs to be run in terms of software, hardware, and data, then we will list the results that we have reached from the experiments done.

4.2 Testing environment

In this section the hardware, software, and data specifications that are needed for the system testing are demonstrated as follows:

4.2.1 Hardware and software specifications

A cluster of machines, as seen in Figure 4.1, are used to run the application, it is heterogeneous environment where three different architectures on seven machines are used. One multi-processors workstation has two Intel dual Xeon CPU 3.06 GHz for each one, with two 512 KB cache memory and 1 GB main memory. Three machines each one has 1.80 GHz Intel core due 2 CPU,1024 KB cache memory and 1 GB main memory. Three machines each one has 3.20 GHz Intel Pentium 4 CPU, 512 KB cache memory and 512 MB main memory. All these machines are connected

via 1 GB switch.

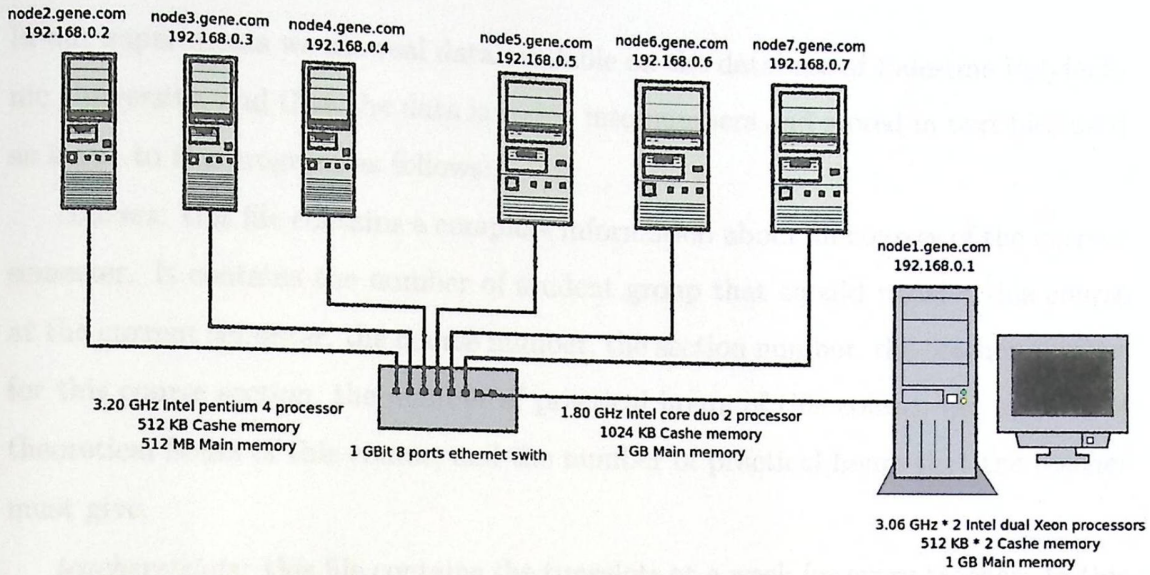


Figure 4.1: Testing environment

The cluster is running on the Linux O.S, fedora 5, and all the applications used in this project are open source software.

All machines are configured to run the algorithm, where all of them have the same user account, and all the users accounts with the same name on all machines has one home directory, on other words, you will find on each machine user called mohammed for example, and mohammed will have shared home directory between all machines via NFS server. In addition to that, all accounts on the cluster can have the right to access any machine from other machines via passwordless SSH service.

At each machine, lam-mpi runtime service was installed, to run the algorithm after including the mpi codes to distribute it.

The development of this algorithm by using Eclipse IDE with mpi-plug-in, and the running of the algorithm application will be from the Linux terminals.

4.2.2 Data specifications

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

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

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

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

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

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

teachersoutput: in this file we determined the timeslots for each teacher, where 0: means free time. 1: means allocated timeslot.

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

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

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

4.3 Experimental results

At this section we will list a series of experiments that show the relationships between variables in diagrams, and analysis of the project outputs.

Figure 4.2 shows the relation between the execution time required to execute the parallel multi-objective algorithm and number of processors, where the population size (popSize= 300) and the number of generations (numGener = 20).

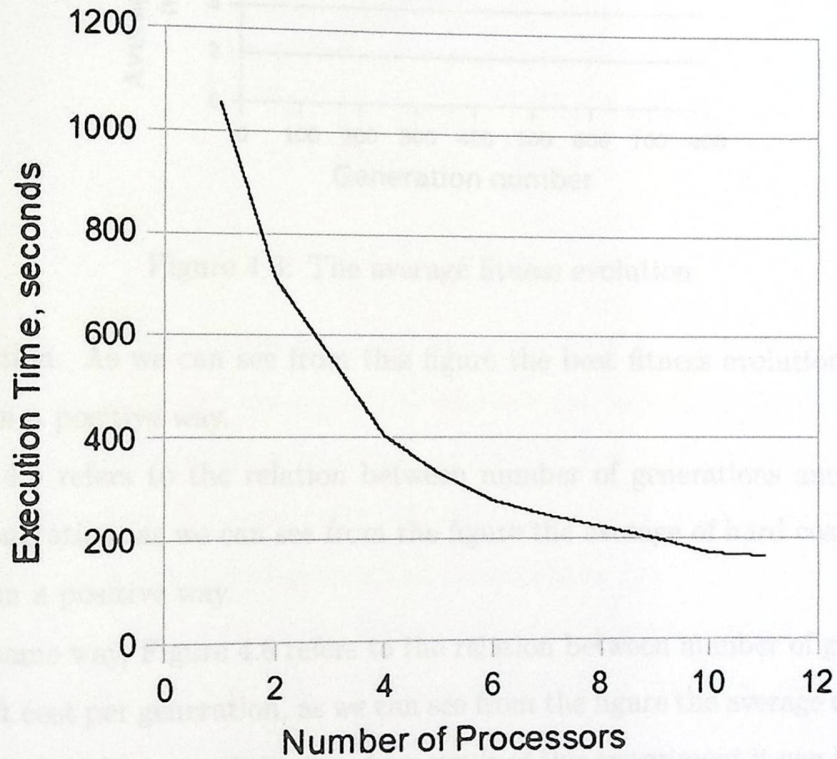


Figure 4.2: The relationship between the execution time and the number of processors

Figure 4.3 shows the relation between number of generation and the average fitness per generation, where the population size (popSize= 700) and the number of generations (numGener = 700). As we can see from this figure the average fitness evolution has been increased in a positive way, which means that the average fitness increased as the processes repeated.

Figure 4.4 shows the relation between number of generation and the best fitness

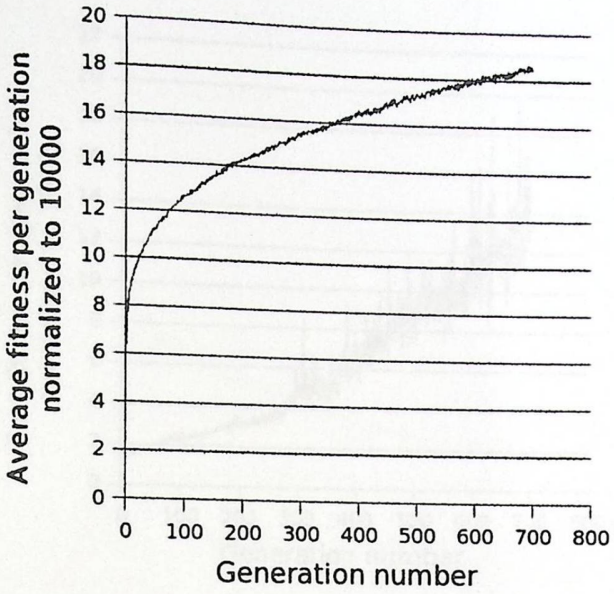


Figure 4.3: The average fitness evolution

per generation. As we can see from this figure the best fitness evolution has been increased in a positive way.

Figure 4.5 refers to the relation between number of generations and the hard cost per generation, as we can see from the figure the average of hard cost has been decreased in a positive way.

In the same way, Figure 4.6 refers to the relation between number of generations and the soft cost per generation, as we can see from the figure the average of soft cost has been decreased in a positive way. As a result of this experiment it can be realized that the multi-objective optimization has been achieved, and both objectives (hard, soft) have been satisfied.

Figure 4.7, and Figure 4.8 shows the relation between the number of generations and both objectives (hard cost, soft cost) it takes 10582 second to generate the solution for population size (popSize= 700) and the number of generations (numGener = 700) on a 11 processors by using the parallel multi-objectives algorithm. We notice in Figure 4.7 that the hard cost has been decreased during the evolution of generations, at the same way the soft cost is decreased too.

Figure 4.9 shows the relation between number of generation and the best fitness

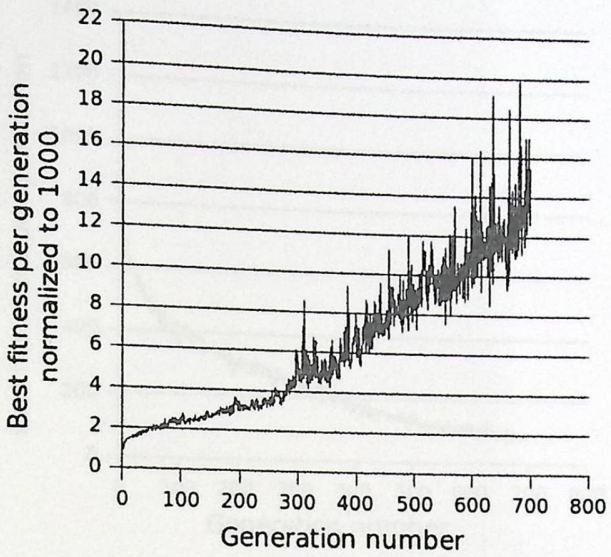


Figure 4.4: The best fitness evolution

per generation. As we can see from this figure the best fitness evolution has been increased in a positive way using parallel multi-objective optimization.

Figure 4.10 represents three experiments for different number of processors and show how that affect the hard cost. At each experiment we use population size (popSize=300) and number of generations (numGener=20).

Figure 4.11 represents the three experiments for different number of processors and how that affect the soft cost. At these experiments, we show that the parallelization process did not affect the soft cost.

In Figure 4.12, we show that the parallelization process did not affect the fitness, we do this experiment on 700 individuals for 700 generations two times for parallel algorithm and 4 times for sequential one.

In Figure 4.13 and (4.14) we take the average of 6 experiments, 2 parallel and 4 sequential to see the relation between best hard cost and number of generations at the first chart and the best soft cost and number of generations at the second. We take the average of all these experiments to make the carve smooth.

Figure 4.15 shows one parallel experiment take the average fitness for each node at this experiment. This experiment use 6 node to implement the algorithm on 300

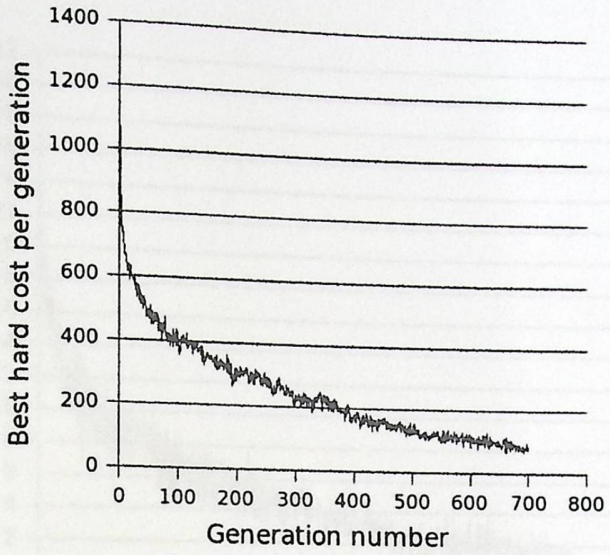


Figure 4.5: The hard cost evolution

individuals for 20 generation.

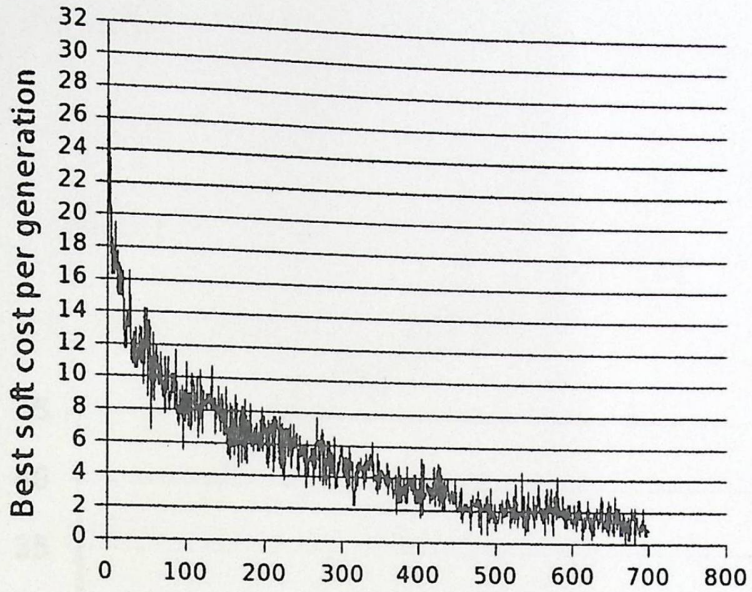


Figure 4.6: The soft cost evolution

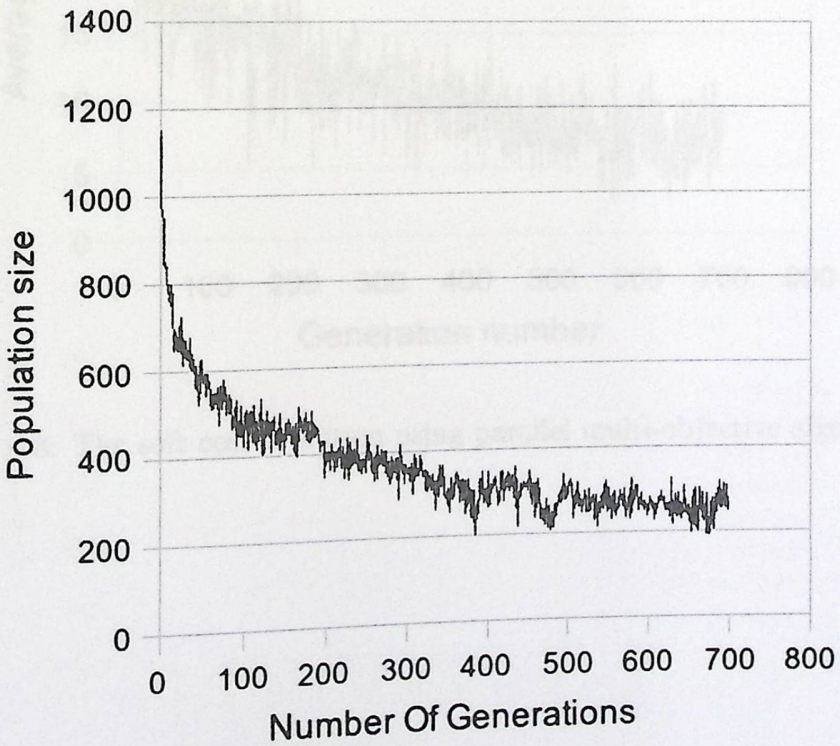


Figure 4.7: The hard cost evolution using parallel multi-objective algorithm

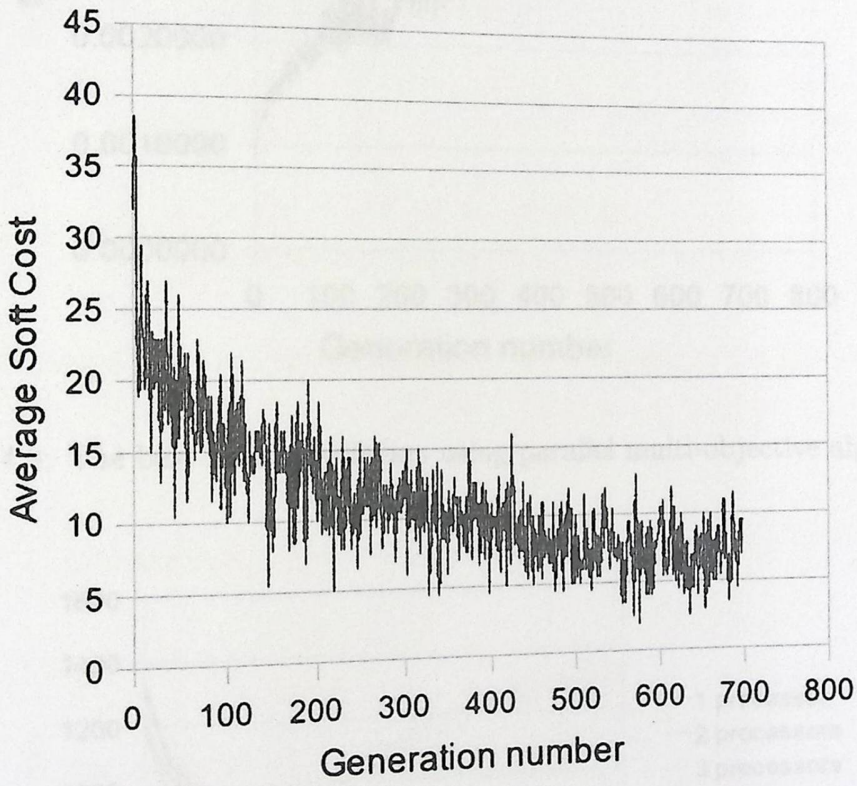


Figure 4.8: The soft cost evolution using parallel multi-objective algorithm

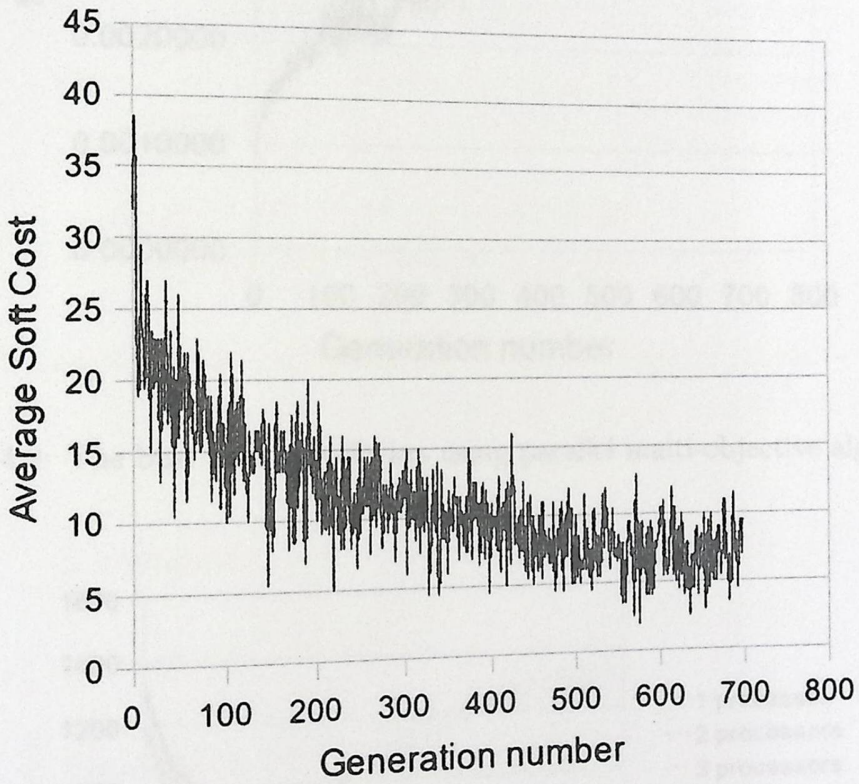


Figure 4.8: The soft cost evolution using parallel multi-objective algorithm

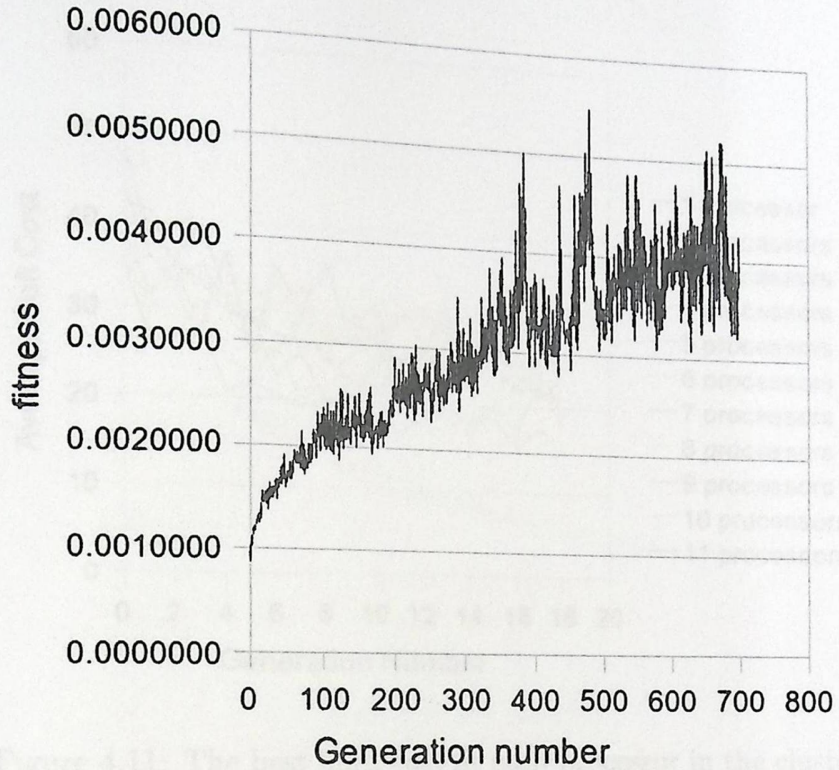


Figure 4.9: The best fitness evolution using parallel multi-objective algorithm

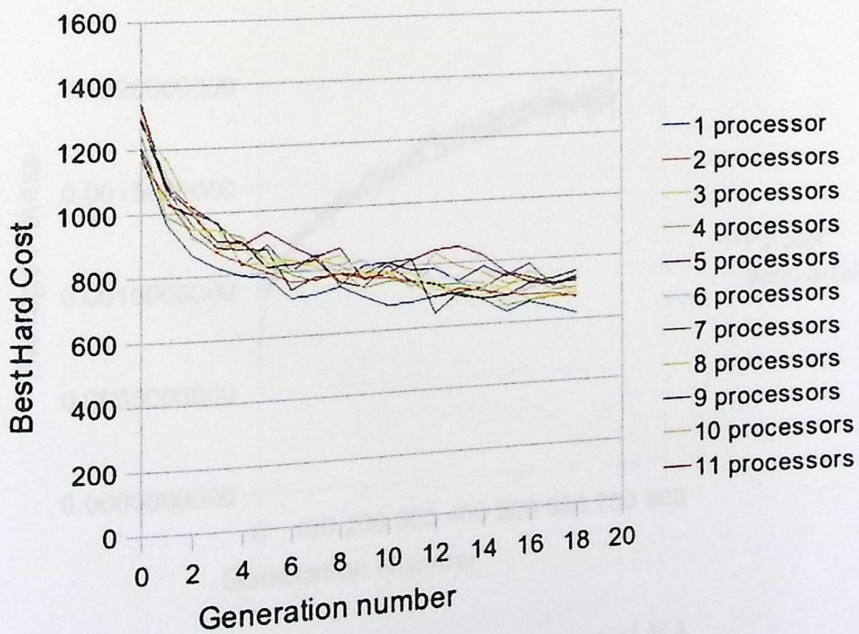


Figure 4.10: The best hard cost at each processor in the cluster

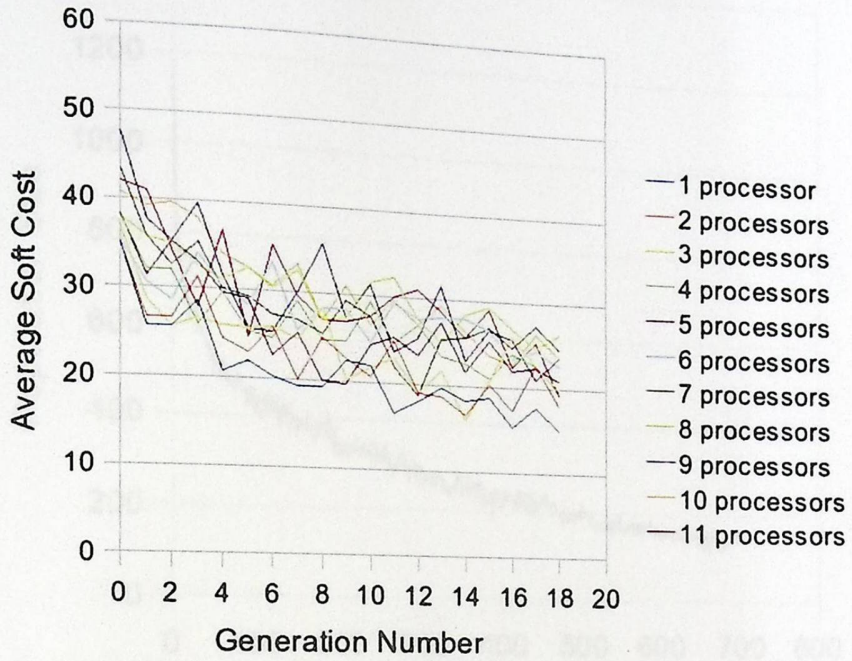


Figure 4.11: The best hard cost at each processor in the cluster

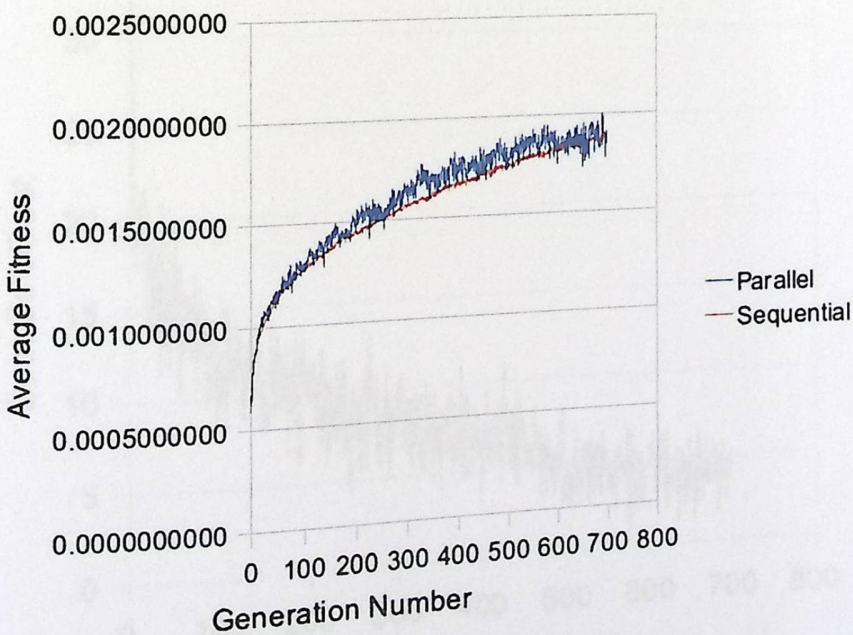


Figure 4.12: Parallel vs sequntial EA.

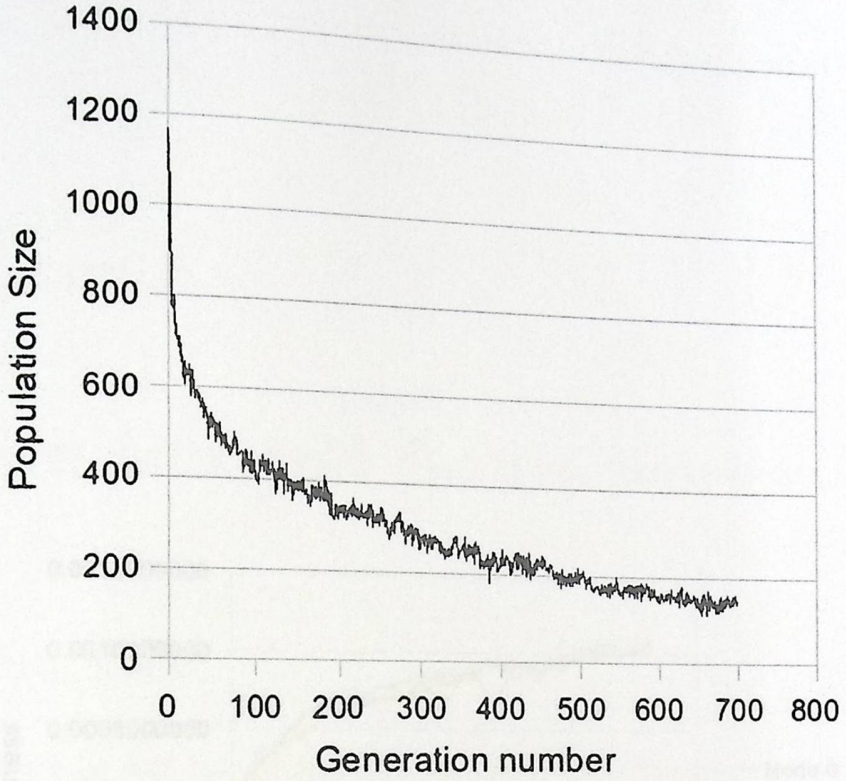


Figure 4.13: Average hard cost from a parallel and sequntial experiments

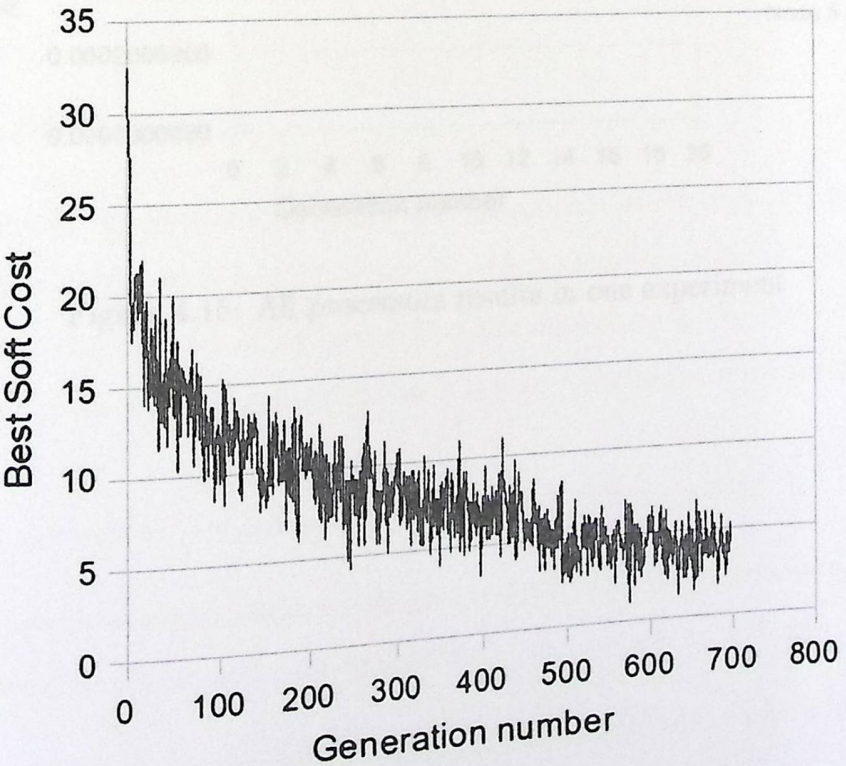


Figure 4.14: Average soft cost from a parallel and sequntial experiments

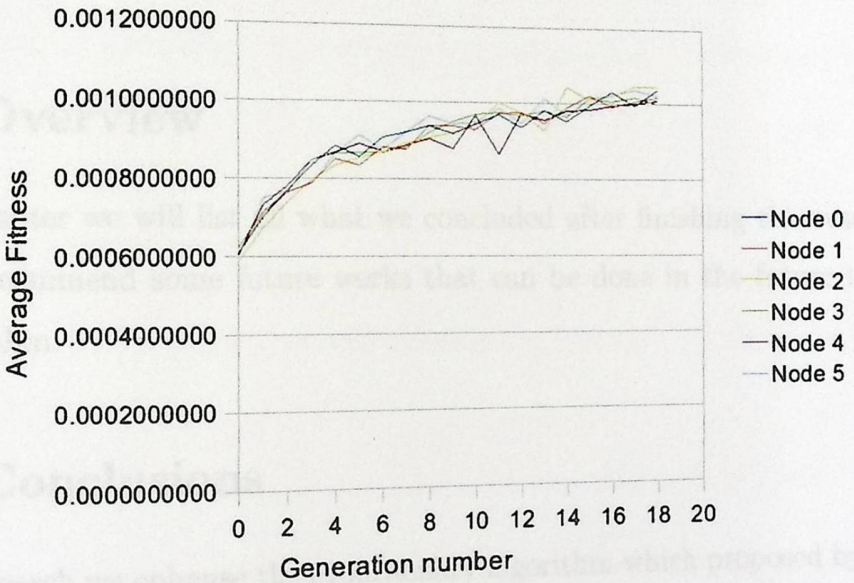


Figure 4.15: All processors results in one experiment

Chapter 5

Conclusion and future works

5.1 Overview

In this chapter we will list all what we concluded after finishing this research, and we will recommend some future works that can be done in the future to enhance the algorithm.

5.2 Conclusions

In this research we enhanse the evolutionary algorithm which proposed by the graduation project in the previous semester [2]. We have reduced the soft cost without affecting the hard cost by using multi-objective optimization. In addition, we have speeded up the algorithm by distributing it among the cluster of machines which we have used.

We can summarize the achieved results in the following points:

1. By implementing the multi-objective and parallel techniques we enhance the proposed model to solve the problem.

2. A multi-objecive technique is proposed with special operations which has improved the results.

3. The proposed methodology is applied on a real data set from the college of administrative sciences and informatics at Palestine Polytechnic University. With all its complexities and constraints.

4. By implementing the parallel algorithm, a small decrease in the fitness of the solutions happened.

5. The communication process consumes small time compared with computational process, most of execution time consumed by mutation function.

6. The load balancing algorithm becomes more complex under the random behaviour of the algorithm, because it is not possible to determine which node will reach the goal faster.

5.3 Future works

After all what we have done, we will mention the following future works to enhance the algorithm:

1. Trying other methods and evolutionary techniques such as PSO algorithm.

2. Considering the problem in the other university colleges. This means a greater and other complex search space.

3. Adding more soft constraints such as the long distances between lectures' rooms as a soft cost.

4. Trying to enhance the load balancing algorithm implemintation for this problem.

5. Using a database management systems instead of files for input and output data.

6. Using the Graphical User Interface (GUI).

Bibliography

[1] A. Abraham, L.Jain, and R.Goldberg, "Evolutionary multiobjective optimization," 2005.

[2] S. Adi, M. Abu-Qopita, and M. Al-Dasht, "University course scheduling using evolutionary algorithms," in *Graduation project, Palestine Polytechnic University*, 2008.

[3] M. Aldasht, M. Alsaheb, S. Adi, and M. A. Qopita, "University course scheduling using evolutionary algorithms," 2009.

[4] M. Aldasht, J. Ortega, and C. Puntonet, "Dynamic load balancing in heterogeneous clusters exploitation of the processing power," 2007.

[5] W. Bozejko and M. Wodecki, "Parallel evolutionary algorithm for the traveling salesman problem," in *Journal of Numerical Analysis, Industrial and Applied Mathematics*, 2007.

[6] L. Cuno, "http://www.klopfenstein.net/lorenz.aspx/genetic-algorithms," *(2009), Genetic Algorithms*, Last accessed 12 Jun 2009.

[7] D. Datta, K. Deb, and C. M. Fonseca, "Solving class timetabling problem of iit kanpur using multi-objective evolutionary algorithm," 2006.

[8] J. Digalakis and K. Margaritis, "Parallel evolutionary algorithms on mpi."

[9] I. Foster, "Designing and building parallel programs," 1995.

[10] S. Ghaemi and M. Vakili, "Using a genetic algorithm optimizer tool to solve university timetable scheduling," 2004.

[11] D. Goldberg, "Genetic algorithms in search, optimization, and machine learning," in *Addison-Wesley: Reading, MA*, 1989.

[12] A. Grama and V. Kumar, "A survey of parallel search algorithms for discrete optimization problems."

[13] M. Lahanas, "http://www.mlahanas.de/moea/mo_optimisation.htm," *(2006), Multiobjective optimization*, Last accessed 17 Jun 2009.

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

[15] M. Opera, "Multi-agent system for universiyt course timetable scheduling," in *The 1st International Conference on Virtual Learning*, 2006, pp. 231–237.

[16] H. Park, A. Grings, M. Santos, and A. Soares, "Parallel hybrid evolutionary computation: Automatic tuning of parameters for parallel gene expression programming," 2008.

[17] P. Pongcharoen, W. Promtetn, P. Yenradee, and C. Hicks, "Stochastic optimisation timetabling tool for university course scheduling," 2007.

[18] P. Pongcharoena, W. Promtet, P. Yenradee, and C. Hicks, "Constructing university timetable using constraint satisfaction programming approach," 2007.

[19] R. Pressman, "Software engineering a practitioner's approach," 2001, pp. 145 – 163.

[20] M. Tomassini, "Parallel and distributed evolutionary algorithms: A review."

[21] L. Zhang and S. Lau, "Constructing university timetable using constraint satisfaction programming approach," 2005.

Appendices

Appendix A

Planing and Analysis

A.1 Planning

This project divided into many task, this dividing base on function and time. Figure (3.14) shows the flow charts of the tasks and who the process done.

Table (A.1) shows the tasks with time needed per week.

A.2 Requirements

In this section we will list the hardware and software requirements, in this project the hardware development requirements and hardware run requirements are the same. Note that the software which used to develop and run this project are open source and free software.

Table (A.2) shows the hardware requirements and its coste, table (A.3) show the software requirements for development and running.

A.3 Gantt chart

This Gantt chart shown in figure (A.1) show the time scheduling for tasks in this project, for 36 weeks (two semesters) all the tasks should be finished with its docu-

Table A.1: Time scheduling table

No.	symbol	breakdown	weeks	notes
1.	T1	Proposal acceptance procedure	1	1^{st} semester
2.	T2	Literature review	10	1^{st}&2^{ed} semester
3.	T3	Review the previous project document and code	4	1^{st} semester
4.	T4	Problem analysis	1	1^{st} semester
5.	T5	Requirements specification and analysis	2	1^{st} semester
6.	T6	Risk analysis	1	1^{st} semester
7.	T7	Analysis and design of the new system for parallel and muti-objective	6	1^{st} semester
8.	T8	Rewriting the code as library files and OOP	3	2^{ed} semester
9.	T9	System implementation and development	6	2^{ed} semester
10.	T10	Unit tests	2	2^{ed} semester
11.	T11	System and integration tests	3	2^{ed} semester
12.	T12	Error correction and debugging	5	2^{ed} semester
13.	T13	Test the system in a real data	1	2^{ed} semester
14.	T14	Analyses the output and results	1	2^{ed} semester
15.	T15	Reporting	*	*

mentation.

A.4 Functional requirements

This project has the following functional requirements:

- Convert the problem to an object oriented.
- Solve the problem using multi-objective optimization to minimize hard and soft constraints.
- Using cluster environment to parallize the execution of the program, in order to make it faster.

Table A.2: Hardware requirements

No.	Item	Specification	Require for	Quantity	Price ($)
1.	Workstation	2 CPU * 3 GHz, 1 GB ram	D and R	1	3000
2.	Dual core	1.6 GHz	D and R	3	1000
3.	Pentium 4	3 GHz	D and R	4	800
4.	Flash memory	1 GB	D and R	1	10
5.	LCD monitor	17 in	D and R	1	200
6.	Mouse		D and R	1	10
7.	keyboard		D and R	1	10
8.	High speed switch	8 ports	D and R	1	50
9.	LAN cables	7 * 3 M	D and R	1	70
				Total	9550

D : Development

R : Running

A.5 Nonfunctional requirements

Utilization: The system should not consumes the hardware capabilities.

Response Time: The system should not take long time to return the results.

Compatibility: The system must run on any platform.

Efficiency: the system must do its job completely.

effectiveness: the system must do its job with better performance than the previous work.

A.6 Risks analysis

In this section, a view of most dangerous risks presented. A template form [19] used to analyze it.

Table A.2: Hardware requirements

No.	Item	Specification	Require for	Quantity	Price ($)
1.	Workstation	2 CPU * 3 GHz, 1 GB ram	D and R	1	3000
2.	Dual core	1.6 GHz	D and R	3	1000
3.	Pentium 4	3 GHz	D and R	4	800
4.	Flash memory	1 GB	D and R	1	10
5.	LCD monitor	17 in	D and R	1	200
6.	Mouse		D and R	1	10
7.	keyboard		D and R	1	10
8.	High speed switch	8 ports	D and R	1	50
9.	LAN cables	7 * 3 M	D and R	1	70
				Total	9550

D : Development

R : Running

A.5 Nonfunctional requirements

Utilization: The system should not consumes the hardware capabilities.

Response Time: The system should not take long time to return the results.

Compatibility: The system must run on any platform.

Efficiency: the system must do its job completely.

effectiveness: the system must do its job with better performance than the previous work.

A.6 Risks analysis

In this section, a view of most dangerous risks presented. A template form [19] used to analyze it.

Table A.3: Software requirements

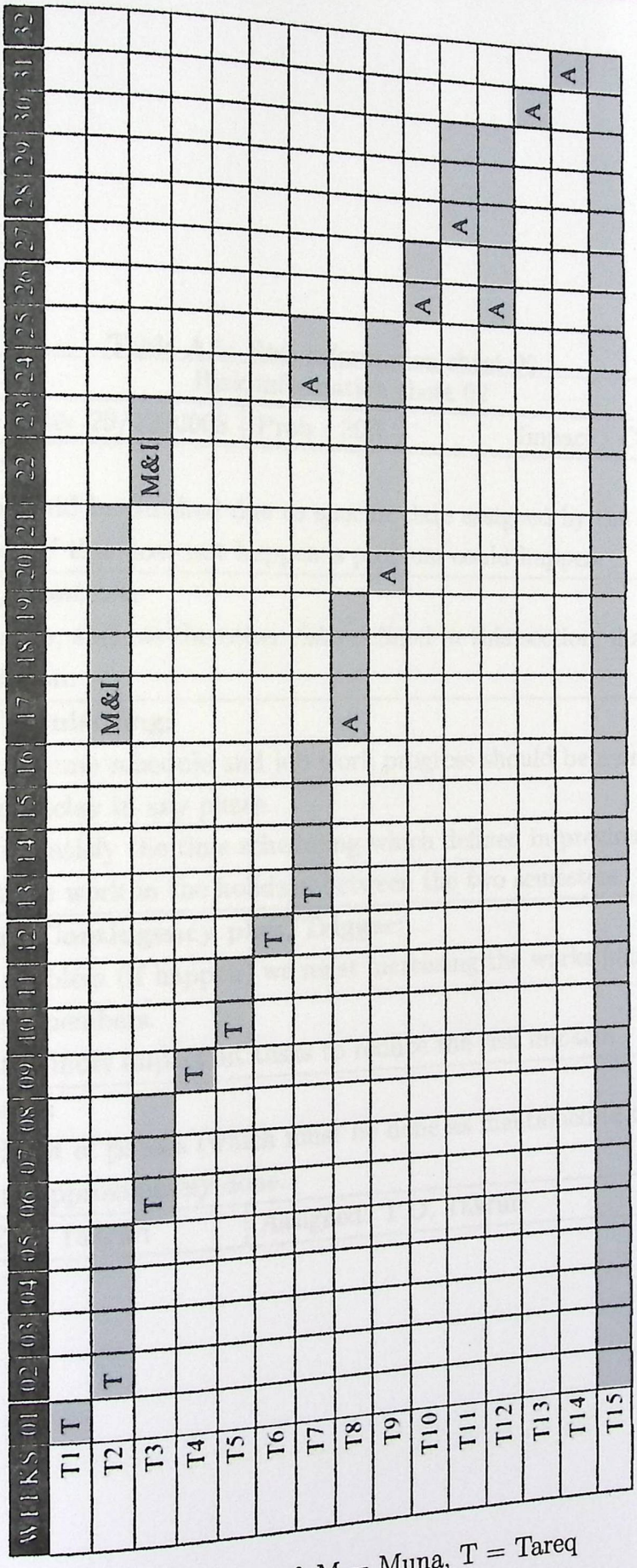
No.	Item	License	Require for	Quantity	Price ($)
1.	Fedora 7 Linux OS	Open Source	D and R	1	0
2.	LAM-MPI Library	Open Source	D	1	0
3.	LAM-MPI runtime	Open Source	R	1	0
4.	Eclipse C/C++	Open Source	D	1	0
5.	Tetex	Open Source	D	1	0
6.	Texmacker - Latex editor	Open Source	D	1	0
7.	Text editor	Open Source	D	1	0
8.	Open office	Open Source	D	1	0
9.	g++ compiler	Open Source	D	1	0
10.	Kpdf - Pdf reader	Open Source	D and R	1	0
11.	Dia	Open Source	D	1	0
				Total	0

D : Development

R : Running

Table A.4: Risk information sheet 01

Risk information sheet 01			
Risk ID: 01	Date: 29/12/2008	Prob : 20%	Impact : Critical
Description: If one or more of the work group quit for any reasons, this will affect on time schedules, increasing group work pressure etc.			
Refinement/Context: Simply, all persons could have emergency situation, specially on our conditions as palestinian. While we have not large number in the work group, loss any one of him may affect critically.			
Mitigation/Monitoring: Each job/tasks should be solved by two person (main researcher and secondary one).			
Management/Contingency plan/Trigger: If any person quit, his/her tasks should be divided to his/her partners who can work it. This must happen until find another person and prepare him			
Current status: 29/12/2008: All membership students are ready to works and with good conditions.			
Originator: T.O. Takruri		Assigned: T.O. Takruri	



A = All, I = Insaf, M = Muna, T = Tareq

Figure A.1: Project Gantti map

Table A.5: Risk information sheet 02

Risk information sheet 02			
Risk ID: 02	Date: 29/12/2008	Prob : 30%	Impact : Critical
Description: This project should be finished due to specific date assigned by the college's administration. If this dose not happen a problem could happen.			
Refinement/Context: For many reasons, such as the other risks defined in this section, may delay the project submission.			
Mitigation/Monitoring: At each week, a time schedule and job work progress should be monitored to check if there are any delay in any phase. We must try to satisfy the time scheduling which defined in previous section. We may start the work in the holidays between the two semesters.			
Management/Contingency plan/Trigger: To solve this problem (if happen) we must increasing the works hour for all of the work group members. Try to finish the most important tasks to reduce the risk impacts			
Current status: 29/12/2008: Most of phases (which must be done as mentioned in the time scheduling) are approximately done.			
Originator: T.O. Takruri		Assigned: T.O. Takruri	

Table A.6: Risk information sheet 03

Risk information sheet 03			
Risk ID: 03	Date: 29/12/2008	Prob : 50%	Impact : Catastrophic
Description: The main aim of this project, is to get best results with least execution time. Most cost may require, so this speedup must be as the increment of the costs to satisfy the best effectiveness. In the worst case, This speedup may not satisfy the best effectiveness which we wish.			
Refinement/Context: The problem may happen if a mistake occur when choosing the techniques that will use to parallize the mention algorithm or when implement it.			
Mitigation/Monitoring: Studying the problem carefully to find best techniques may reduce this risk impact and probability.			
Management/Contingency plan/Trigger: To try to use other techniques for parallization. And to study the other solutions and arguments which may affect on the effectiveness.			
Current status: 29/12/2008: Studying the possible techniques available in the previous studies to find the best.			
Originator: T.O. Takruri		Assigned: T.O. Takruri	

Table A.6: Risk information sheet 03

Risk information sheet 03			
Risk ID: 03	Date: 29/12/2008	Prob : 50%	Impact : Catastrophic
Description: The main aim of this project, is to get best results with least execution time. Most cost may require, so this speedup must be as the increment of the costs to satisfy the best effectiveness. In the worst case, This speedup may not satisfy the best effectiveness which we wish.			
Refinement/Context: The problem may happen if a mistake occur when choosing the techniques that will use to parallize the mention algorithm or when implement it.			
Mitigation/Monitoring: Studying the problem carefully to find best techniques may reduce this risk impact and probability.			
Management/Contingency plan/Trigger: To try to use other techniques for parallization. And to study the other solutions and arguments which may affect on the effectiveness.			
Current status: 29/12/2008: Studying the possible techniques available in the previous studies to find the best.			
Originator: T.O. Takruri		Assigned: T.O. Takruri	

Table A.7: Risk information sheet 04

Risk information sheet 04			
Risk ID: 04	Date: 29/12/2008	Prob : 20%	Impact : low
Description: One or more of clustered machine, networks devices or the software many corrupted, in other words the development environment may loss some parts.			
Refinement/Context: The hardware may damage, and the software may has a bugs. Many dangers affects on hardware and software such as electricity power changes rapidly, system misuse, data loss and others.			
Mitigation/Monitoring: A full backup shall do every week or when a large change happen. Alternative devices may be spar and ready to use in the emergency situation.			
Management/Contingency plan/Trigger: While we have a backup, if a software corrupted occur, a restore must done to recover it. If it is a hardware corrupted, it will be one of these two cases. The first case if it is secondary parts we many chance the system to works without it, the other case, if it is main part an alternative device must exists to replace it.			
Current status: 29/12/2008: All the machines and devices already prepared with good quality to develop and run the system on it.			
Originator: T.O. Takruri		Assigned: T.O. Takruri	

Appendix B

Time table

Appendix B

Time table

Appendix B

Time table

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

الخميس	الاربعاء	الثلاثاء	الاثنين	الاحد	قاعة	الشعبة	س.م	رقم المساق	اسم المساق
8 – 9		8 – 9		8 – 9	325	3	3	5052	لغة انجليزية استدراكي
		10 – 11		10 – 11	312	6	3	5055	الحاسوب واساسيات البرمجة
			11 – 2		PC2				
11 – 12		11 – 12		11 – 12	110	4	3	4001	اللغة العربية
12 – 1:15				12 – 1:15	222	4	3	5054	استدراكي حاسوب
3 – 4		3 – 4		3 – 4	223	2	3	5029	مقدمة في الاحصاء
2 – 3		2 – 3		2 – 3	110	1	3	5029	مقدمة في الاحصاء
	3:30 -5		3:30 -5		312	1	3	4247	الاقتصاد الجزئي
3 – 4		3 – 4		3 – 4	110	2	3	4247	الاقتصاد الجزئي
4 – 5		4 – 5		4 – 5	310	5	3	4502	مبادئ محاسبة 1
	8 – 9:30		8 – 9:30		325	8	3	4070	لغة انجليزية 2
	9:30 – 11		9:30 – 11		312	1	3	4541	مقدمة في نظم المعلومات
	2 – 3:30		2 – 3:30		223	1	3	4503	مقدمة في الادارة
	3:30 -5		3:30 -5		223	9	3	4003	لغة انجليزية 1
2 – 3		2 – 3		2 – 3	320	10	3	4003	اللغة الانجليزية 1

Information Systems second level

الخميس	الاربعاء	الثلاثاء	الاثنين	الاحد	قاعة	الشعبة	س.م	رقم المساق	اسم المساق
9 – 10		9 – 10		9 – 10	110	1	3	4265	السلوك التنظيمي
	8 – 9:30		8 – 9:30		214	1	3	4257	مبادئ تنظيم وعمارة الحاسوب
10 – 11		10 – 11		10 – 11	218	2	3	4257	مبادئ تنظيم وعمارة الحاسوب
11 – 12				11 – 12	213	1	3	4823	برمجة الحاسوب
2 – 5					PC3				
12 – 1				12 – 1	110	4	2	4015	اساليب البحث العلمي
	8 – 9:30		8 – 9:30		214	1	3	4824	تركيب بيانات
		2 – 5			PC2				
		8 – 9		8 – 9	403	3	3	4824	تركيب بيانات
				2 – 5	PC2				
	9:30 – 11		9:30 – 11		319	1	3	5035	المحاسبة الادارية
	11 – 12:30		11 – 12:30		403	1	3	4259	مقدمة في نظم التشغيل
	12:30 – 2		12:30 – 2		223	1	3	4543	مبادئ التسويق
	2 – 3:30		2 – 3:30		319	1	3	5016	تحليل النظم

Information Systems fourth level

Appendix B

Time table

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

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
لغة انجليزية استدراكي	5052	3	3	325	8 – 9		8 – 9		8 – 9
الحاسوب واساسيات البرمجة	5055	3	6	312	10 – 11		10 – 11		
				PC2		11 – 2			
اللغة العربية	4001	3	4	110	11 – 12		11 – 12		11 – 12
استدراكي حاسوب	5054	3	4	222	12 – 1:15				12 – 1:15
مقدمة في الاحصاء	5029	3	2	223	3 – 4		3 – 4		3 – 4
مقدمة في الاحصاء	5029	3	1	110	2 – 3		2 – 3		2 – 3
الاقتصاد الجزئي	4247	3	1	312		3:30 -5		3:30 -5	
الاقتصاد الجزئي	4247	3	2	110	3 – 4		3 – 4		3 – 4
مبادئ محاسبة 1	4502	3	5	310	4 – 5		4 – 5		4 – 5
لغة انجليزية 2	4070	3	8	325		8 – 9:30		8 – 9:30	
مقدمة في نظم المعلومات	4541	3	1	312		9:30 – 11		9:30 – 11	
مقدمة في الادارة	4503	3	1	223		2 – 3:30		2 – 3:30	
لغة انجليزية 1	4003	3	9	223		3:30 -5		3:30 -5	
اللغة الانجليزية 1	4003	3	10	320	2 – 3		2 – 3		2 – 3

Information Systems second level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
السلوك التنظيمي	4265	3	1	110	9 – 10		9 – 10		9 – 10
مبادئ تنظيم وعمارة الحاسوب	4257	3	1	214		8 – 9:30		8 – 9:30	
مبادئ تنظيم وعمارة الحاسوب	4257	3	2	218	10 – 11		10 – 11		10 – 11
برمجة الحاسوب	4823	3	1	213	11 – 12				11 – 12
				PC3					2 – 5
اساليب البحث العلمي	4015	2	4	110	12 – 1				12 – 1
تركيب بيانات	4824	3	1	214		8 – 9:30		8 – 9:30	
				PC2			2 – 5		
تركيب بيانات	4824	3	3	403	8 – 9		8 – 9		
				PC2	2 – 5				
المحاسبة الادارية	5035	3	1	319		9:30 – 11		9:30 – 11	
مقدمة في نظم التشغيل	4259	3	1	403		11 – 12:30		11 – 12:30	
مبادئ التسويق	4543	3	1	223		12:30 – 2		12:30 – 2	
تحليل النظم	5016	3	1	319		2 – 3:30		2 – 3:30	

Information Systems fourth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
نظم ادارة قواعد بيانات	4255	3	1	320	10 – 11		10 – 11		10 – 11
مقدمة في التجارة الالكترونية	4642	3	3	310	12 – 1:15				12 – 1:15
نظم المعلومات الادارية 1	4261	3	1	218	2 – 3		2 – 3		2 – 3
ادارة التسويق	4270	3	1	312		8 – 9:30		8 – 9:30	
لادارة الاستراتيجية واتخاذ القرارات	4268	3	1	110		11 – 12:30		11 – 12:30	
الاقتصاد الكلي	4248	3	1	221		12:30 – 2		12:30 – 2	

Information Systems sixth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مقدمة في ادارة المشاريع	4281	3	1	319		8 – 9:30		8 – 9:30	
تاريخ فلسطين الحديث	4320	3	4	110		9:30 – 11		9:30 – 11	

Information Systems eighth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مقدمة في الادارة	4503	3	3	325	11 – 12		11 – 12		11 – 12
مقدمة في الادارة	4503	3	2	214		9:30 – 11		9:30 – 11	
الحاسوب واساسيات البرمجة	5055	3	7	214	12 – 1:15				12 – 1:15
				PC1					8 – 11
اقتصاد جزئي	4247	3	4	319		11 – 12:30		11 – 12:30	
اقتصاد جزئي	4247	3	3	222	2 – 3		2 – 3		2 – 3
لغة انجليزية 2	4070	3	11	319	3 – 4		3 – 4		3 – 4
ثقافة اسلامية	4002	3	5	110		8 – 9:30		8 – 9:30	
مبادئ محاسبة 2	4246	3	2	319		11 – 12:30		11 – 12:30	
مبادئ محاسبة 2	4246	3	1	312		3:30 -5		3:30 -5	
مقدمة في الاحصاء	5029	3	3	325		12:30 – 2		12:30 – 2	
لغة انجليزية 1	4003	3	11	312		2 – 3:30		2 – 3:30	

Administrative management second level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
اساسيات ابرمجة	4819	3	1	312	4 – 5				4 – 5
				PC3				2 – 5	
اساسيات ابرمجة	4819	3	2	312			8 – 9		8 – 9
				PC4	8 – 11				
الثقافة الاسلامية	4002	3	7	310	11 – 12		11 – 12		11 – 12
ادارة العمليات	4267	3	1	310	10 – 11		10 – 11		10 – 11
ادارة العمليات	4267	3	2	222	12 – 1:15				12 – 1:15
مراسلات الاعمال	4544	3	2	222	3 – 4		3 – 4		3 – 4
مراسلات الاعمال	4544	3	1	221	2 – 3		2 – 3		2 – 3
المحاسبة الادارية	4264	3	1	320		8 – 9:30		8 – 9:30	
مقدمة في نظم المعلومات	4541	3	3	223		11 – 12:30		11 – 12:30	
اتمتة مكاتب	4822	2	2	312					10 – 11
				PC1		2 – 5			
اتمتة مكاتب	4822	2	1	213	2 – 3				
				PC3		2 – 5			

Administrative management fourth level

اسم المساق	رقم المساق	مركز	الشعبة	الساعة
البرمجة المرئية لطلبة الادارة	4625	3	2	310
البرمجة المرئية لطلبة الادارة	4625	3	1	310
نظم المعلومات الادارية 1	4261	3	2	312
ادارة المشاريع الصغيرة	4289	3	1	312
الادارة المحلية	4576	3		310
نظم المعلومات الادارية 2	4262	3	1	218
ادارة الشراء والمخازن	4567	3	2	221
ادارة الشراء والمخازن	4567	3	1	110

الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
		10 – 11		10 – 11
				12 – 1:15
		2 – 3		2 – 3
	8 – 9:30		8 – 9:30	
	11 – 12:30		11 – 12:30	
	12:30 – 2		12:30 – 2	

الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
	8 – 9:30		8 – 9:30	
	9:30 – 11		9:30 – 11	

اسم المساق	رقم المساق	مركز	الشعبة	الساعة
قانون اعمال	4271	3	1	310
اخلاقيات الاعمال والمسؤولية	4556	3	1	222
العلاقات العامة	4565	3	1	218
ادارة الاعمال العالمية	4558	3	1	221
اللغة العبرية	4308	3	3	110
حلقة بحث	4644	1	1	213

الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
		11 – 12		11 – 12
	9:30 – 11		9:30 – 11	
				12 – 1:15
				8 – 11
			11 – 12:30	
		2 – 3		2 – 3
		3 – 4		3 – 4
	8 – 9:30		8 – 9:30	
	11 – 12:30		11 – 12:30	
	3:30 -5		3:30 -5	
	12:30 – 2		12:30 – 2	
	2 – 3:30		2 – 3:30	

الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
				4 – 5
			2 – 5	
		8 – 9		8 – 9
		11 – 12		11 – 12
		10 – 11		10 – 11
				12 – 1:15
		3 – 4		3 – 4
		2 – 3		2 – 3
			8 – 9:30	
			11 – 12:30	
				10 – 11

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
البرمجة المرئية لطلبة الادارة	4625	3	2	213			9 – 10		9 – 10
				PC2		11 – 2			
البرمجة المرئية لطلبة الادارة	4625	3	1	320	8 – 9		8 – 9		
				PC1				2 – 5	
نظم المعلومات الادارية 1	4261	3	2	222	11 – 12		11 – 12		11 – 12
ادارة المشاريع الصغيرة	4269	3	1	312	12 – 1:15				12 – 1:15
الادارة المحلية	4576	3	1	110	4 – 5		4 – 5		4 – 5
نظم المعلومات الادارية 2	4262	3	1	218		8 – 9:30		8 – 9:30	
ادارة الشراء والمخازن	4567	3	2	221		9:30 – 11		9:30 – 11	
ادارة الشراء والمخازن	4567	3	1	110	2 – 3		2 – 3		2 – 3

Administrative management sixth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
قانون اعمال	4271	3	1	310	2 – 3		2 – 3		2 – 3
اخلاقيات الاعمال والمسؤولية	4556	3	1	222	3 – 4		3 – 4		3 – 4
العلاقات العامة	4565	3	1	218		9:30 – 11		9:30 – 11	
ادارة الاعمال العالمية	4558	3	1	221		11 – 12:30		11 – 12:30	
اللغة العبرية	4308	3	3	110		2 – 3:30		2 – 3:30	
حلقة بحث	4644	1	1	213			10 – 11		

Administrative management eighth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
اللغة العربية	4001	3	5	110		12:30 – 2		12:30 – 2	
اللغة العربية	4001	3	6	110	8 – 9		8 – 9		8 – 9
اللغة الانجليزية 1	4003	3	13	325	9 – 10		9 – 10		9 – 10
مقدمة في الادارة	4503	3	4	319	11 – 12		11 – 12		11 – 12
الحاسوب واساسيات البرمجة	5055	3	8	403	2 – 3		2 – 3		
				PC1				8 – 11	
تفاضل وتكامل 2	4005	3	15	218		9:30 – 11		9:30 – 11	
تفاضل وتكامل 2	4005	3	14	218		11 – 12:30		11 – 12:30	
تفاضل وتكامل 2	4005	3	13	223	11 – 12		11 – 12		11 – 12
مختبر نظم التشغيل	4892	1	2	PC1					2 – 5
مختبر نظم التشغيل	4892	1	1	PC2				2 – 5	
مختبر نظم التشغيل	4892	1	1	PC3				11 – 2	
مختبر نظم التشغيل	4892	1	1	PC2				2 – 5	
برمجة الحاسوب	4823	3	4	221	8 – 9		8 – 9		
				PC2		2 – 5			
برمجة الحاسوب	4823	3	2	218		2 – 3:30		2 – 3:30	
				PC1	2 – 5				
برمجة الحاسوب	4823	3	3	403	9 – 10		9 – 10		
				PC3		8 – 11			
مقدمة في الاحصاء	4507	3	3	403		8 – 9:30		8 – 9:30	
اللغة الانجليزية 2	4070	3	10	221	10 – 11		10 – 11		10 – 11
تفاضل وتكامل 1	4004	3	5	403		9:30 – 11		9:30 – 11	
مبادئ المحاسبة 1	4502	3	2	312		11 – 12:30		11 – 12:30	
مبادئ المحاسبة 1	4502	3	3	310		3:30 -5		3:30 -5	
مبادئ المحاسبة 1	4502	3	1	310		2 – 3:30		2 – 3:30	

Information Technology second level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مقدمة في نظم المعلومات	4541	3	4	320	8 – 9		8 – 9		8 – 9
الثقافة الاسلامية	4002	3	6	310	9 – 10		9 – 10		9 – 10
الاقتصاد الكلي	4248	3	2	310	10 – 11		10 – 11		10 – 11
البرمجة المرئية	4542	3	2	218	12 – 1				12 – 1
				PC1					2 – 5
البرمجة المرئية	4542	3	1	213		2 – 3:30		2 – 3:30	
				PC4			2 – 5		
هندسة برمجيات	4256	3	2	310		9:30 – 11		9:30 – 11	
هندسة برمجيات	4256	3	3	218		12:30 – 2		12:30 – 2	
مبادئ الرسم الحاسوبي	4274	3	1	213		9:30 – 11		9:30 – 11	
				PC3		11 – 2			
مبادئ الرسم الحاسوبي	4274	3	2	223	12 – 1				12 – 1
				PC4		11-Aug			
مقدمة في نظم التشغيل	4259	3	3	403		12:30 – 2		12:30 – 2	
مقدمة في نظم التشغيل	4259	3	2	221		3:30 -5		3:30 -5	
السلوك التنظيمي	4265	3	2	222		2 – 3:30		2 – 3:30	
السلوك التنظيمي	4265	3	3	221	11 – 12		11 – 12		11 – 12

Information Technology fourth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مبادئ التسويق	4543	3	2	223	8 – 9		8 – 9		8 – 9
الحقيقة الافتراضية	4620	3	2	222	9 – 10				9 – 10
				PC4		2 – 5			
الحقيقة الافتراضية	4620	3	1	223	10 – 11				10 – 11
				PC4				8 – 11	
تصميم تطبيقات الانترنت	4548	3	1	214	11 – 12				11 – 12
				PC3	2 – 5				
شبكات وتراسل بيانات	4260	3	2	213	12 – 1:15				12 – 1:15
شبكات وتراسل بيانات	4260	3	1	310		11 – 12:30		11 – 12:30	
برمجة كيانات	4614	3	2	403			10 – 11		10 – 11
				PC2		8 – 11			
برمجة كيانات	4614	3	1	410	12 – 1				12 – 1
				PC4					2 – 5
برمجة قواعد بيانات	4275	3	1	320		11 – 12:30		11 – 12:30	
				PC3				8 – 11	
برمجة قواعد بيانات	4275	3	2	319	9 – 10				9 – 10
				PC1		8 – 11			
مقدمة في التجارة الالكترونية	4642	3	2	319		12:30 – 2		12:30 – 2	

Information Technology sixth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مقدمة في معالجة الصور	4617	3	1	312	11 – 12		11 – 12		11 – 12
تاريخ فلسطين الحديث	4320	3	3	403	3 – 4		3 – 4		3 – 4
البرمجة للانترنت	4615	3	1	319			10 – 11		10 – 11
				PC2				8 – 11	
هندسة برمجيات متقدمة	4550	3	1	410		11 – 12:30		11 – 12:30	
اللغة الفرنسية	4323	3	2	221		2 – 3:30		2 – 3:30	
حلقة البحث	4279	1	1	213		3:30 -5			

Information Technology eighth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
اللغة الانجليزية 1	4003	3	14	222	8 – 9		8 – 9		8 – 9
مقدمة في الادارة	4503	3	5	218	9 – 10		9 – 10		9 – 10
اساسيات تصميم 1	4731	3	1	213	10 – 11				10 – 11
				مشغل	2 – 5				
الالوان ومزجها	4745	2	1	مشغل		8 – 11		8 – 11	
علم نفس	4251	3	1	325		11 – 12:30		11 – 12:30	
الرسم الحاسوبي	5146	3	1	312				12:30 – 1:30	
				ملتميديا			2 – 5		2 – 5
الرسم الحاسوبي	5146	3	2	213		12:30 – 1:30			
				ملتميديا		2 – 5		2 – 5	

Graphics and multimedia second level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
مبادئ ابداع	4735	3	1	310	8 – 9		8 – 9		
				مشغل	11 – 2				
مبادئ الرسم الحاسوبي	4274	3	4	312	9 – 10				9 – 10
				PC2					11 – 12
مبادئ المحاسبة 1	4502	3	4	325	10 – 11		10 – 11		10 – 11
البرمجة المرئية	4542	3	3	320		9:30 – 11		9:30 – 11	
				PC2	2 – 5				
معالجة تصميم 1	4732	2	1	ملتميديا		11 – 2		11 – 2	
مقدمة في نظم المعلومات	4541	3	3	320		2 – 3:30		2 – 3:30	

Graphics and multimedia fourth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
البرمجة للوسائط المتعددة	4738	3	1	218	8 – 9		8 – 9		
				ملتميديا					11 – 2
مبادئ التسويق	4543	3	3	223	9 – 10		9 – 10		9 – 10
تصميم الوسائط الرقمية	4741	3	1	110	10 – 11		10 – 11		
				ملتميديا		8 – 11			
قانون الطباعة والنشر	4747	3	1	310		12:30 – 2		12:30 – 2	

Graphics and multimedia sixth level

اسم المساق	رقم المساق	س.م	الشعبة	قاعة	الاحد	الاثنين	الثلاثاء	الاربعاء	الخميس
تصميم الافلام والرسوم المتحركة	4752	3	1	213	9 – 10		9 – 10		
				ملتميديا	11 – 2				
مواضيع خاصة	4755	3	1	410		3:30 -5		3:30 -5	
				ملتميديا					8 – 11
تصميم صفحات الانترنت	4643	2	1	214			11 – 12		
				PC4				11 – 2	

Graphics and multimedia eighth level