# A Multi-Stage Co-Evolutionary Algorithm for a University Timetabling Problem

**Chan Chee Keong**

**School of Electrical & Electronic Engineering**

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

**2006**

# Statement of Originality

I hereby certify that the content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

1 Feb 2006
---
Date


---
Chan Chee Keong

## Acknowledgements

I would like to express my great appreciation to my supervisors A/P Gooi Hoay Beng and A/P Lim Meng Hiot for their invaluable help and guidance throughout the course of my PhD endeavor.

I would like to thank Prof Alex Kot, Head of Information Engineering Division, EEE, for his encouragement and support.

I would also like to thank my wife Dorothy for her constant and unwavering support, love and encouragement.

# Table of Content

## Summary

Timetabling in a university environment is generally regarded as a NP-complete type of problems. Such problems can be best handled using a search algorithm augmented with some form of heuristics. Recent research works in this area have explored the use of evolutionary algorithms as a promising technique to handle the complexity of the problem.

The main focus of this thesis is to address the problem of timetabling faced in a university environment. There are several types of timetables used in such an environment, the main ones being examination and curriculum timetables. Evolutionary algorithms reported in the literature usually focus on scheduling examination timetables. There is less emphasis on curriculum timetabling.

There are two main parts in this thesis. The first part examines the university timetabling problem in details. Special focus is directed at curriculum timetabling. This is achieved by giving more emphasis on deriving a proper formulation of the problem.

The second part examines current techniques for timetabling. The pros merits of these techniques are discussed. Based on these findings, a multi-stage co-evolutionary algorithm (MSCOA) is proposed. This algorithm also uses a novel mutation technique to evolve solutions.

The timetables considered are based on the undergraduate courses of the School of Electrical and Electronic Engineering (EEE) in Nanyang Technological University (NTU). Three types of timetables from EEE are used. They are the examination timetable, curriculum timetable and the subject allocation system. Three case studies, based on these timetables, described in this thesis demonstrate the performance of the proposed evolutionary approach.

From the three case studies, it has been found that the proposed algorithm is both effective and efficient in solving the complex timetabling problems faced by a university. The case of subject allocation had been on trial for two semesters on a controlled group of students. The feedbacks from students have been positive and most favored the use of this system. Plans are on the way to carry out similar trial runs for the examination timetable.

# List of Figures

# List of Tables

# Chapter 1 Introduction

## 1.1. Motivation

Although timetables or schedules are used by many people and organizations, many of these timetables are planned manually. The process is tedious and the planners tend to adopt the first feasible timetable without considering its quality. There are many reasons for this. One of the reasons is that the problem is NP-complete, which necessitates the use of a search or exploratory paradigm. This approach generally requires a significant amount of computing power. Only until recently, research interest in this area has begun to grow. Hence, this is a relatively challenging new area.

Secondly, timetabling in a university environment poses an even greater challenge. The main reasons are the continuing changes in the course curriculum and an ever increasing intake of students in most university environment. In addition, many of the recent research works in university timetabling are focused on examination timetabling. Hence, this thesis work aims to cover all the various types of university timetabling, including the more difficult curriculum timetabling. A major consideration is to develop an approach that can be applied on all of these.

A major drive of this thesis work is an urgent need to automate the planning of the various timetables used in the School of Electrical and

Electronic Engineering (EEE) in Nanyang Technological University (NTU). A successful completion will contribute greatly in alleviating the immense difficulties of allocating the severely limited teaching resources in EEE.

In summary, there are a few impetuses for this thesis:

- Timetabling is a relatively challenging area, especially the timetabling problem of a university
- Most of the research works on university timetabling focus on examination timetabling. Hence, there is a need to explore into other types of university timetables, especially the curriculum timetable.
- This thesis work has significant practical value, especially for university with a large intake of students reading a particular program such as that of NTU.

## 1.2. Objectives

The main objectives of this thesis work are:

- To study the complexity and then define the university timetabling problem statement
- To develop a novel evolutionary algorithm

## 1.3. Major contributions

They are:

- A comprehensive definition of the curriculum timetabling problem in a university environment, which aids in the development of a novel algorithm
- A novel evolutionary algorithm, which can be easily adapted, to solve the various types of a university timetabling problems

In the course of this thesis work, two pilot runs on the students' registration of subjects had been successfully carried out, using the Subject Allocation System. The first pilot run was for the 77 second year ABP (Accelerated Bachelor Program) students of Academic Year 2003/04, Semester 2. The second run was for a volunteer group of 90 third year students taken from Academic Year 2004/05. The majority of the students who participated in these runs favored this new system. They found it to be a fairer, more relaxed system and recommended that it should be adopted widely.

The detailed analysis and development of a novel evolutionary algorithm for a university timetabling system provides a framework as well as laid the foundation for future timetabling works in EEE. What has been developed in this thesis is the most difficult part of an automated timetabling system. The author intends to continue to work on the overall development of a completely automated timetabling system, even after completion of his PhD work.

## 1.4. Review of Recent Works

Timetables are used widely in many organizations to schedule or organize the usage of some limited resources within a period of time. Timetables have been designed to schedule jobs in a machine shop, roster duty for nurses in a hospital, assign bus routes for bus drivers of a bus company, schedule classes in a public school and many other similar applications [1, 2, 3, 4, 5]. In many of these organizations, an optimally planned timetable is crucial to the smooth running of their operations. Hence, the need to define and plan a good timetable is necessary for these companies. Many techniques have evolved to meet this need. Most of these are specifically tailored to the intended applications.

A timetable usually depicts as a table of rows and columns, a plan or schedule concerning the usage of resources [4, 11, 12] to achieve an objective. The most important and constraining resource is the availability of timeslots. Each timeslot is of a fixed duration. This is usually arranged as a consecutive array. A timeslot can be a fixed number of hours, days or even weeks depending on the application. In the case of a university curriculum timetable, a timeslot is usually an hour period within a weekly schedule [4]. The second type of resource is the set of teaching facilities or venues, such as lecture theatres and tutorial rooms, available in each timeslot. Finally, we have the set of teaching activities or classes to be assigned to the set of timeslots.

Associated with each class are the teaching venue and a group of students.

There are many types of timetables used by a university [13, 14, 15, 16]. However, two of them are more prevalent than the others. These are the examination and curriculum timetable. With this broad categorization, the current practice is to approach the two problems separately. Although these timetables receive wide attention, there are other types of problem that bear similar structure.

The task of a timetable planner is to schedule the set of teaching activities completely into the array of timeslots. The planning process is guided by an objective. In addition, there are constraints imposed on the system. These constraints are classified as hard and soft constraints [17, 18]. A solution is termed as feasible if it does not violate any hard constraint. Soft constraints are actually preferences and may be violated. The amount of violation is frequently used to distinguish the desirability or goodness of a solution. The complexity of a timetable system depends on the number of timeslots, the amount and type of teaching activities, availability of teaching facilities and the number of students.

In simple cases, the planning can be done manually by one person, but the process can be very tedious and time-consuming. The planner is

usually satisfied with the first complete feasible timetable, albeit with some fine-tuning. However, the eventual timetable may not be optimal.

Alternatively, the planner can use some standard office software and special programming techniques. Unlike the manual cases, it is now possible for the planner to plan several timetables and to choose the best among them. The aim of such techniques is to produce a timetable that will meet its objective and satisfy efficiently as many constraints as possible. Examples of such techniques include the constraint programming and conventional optimization techniques.

A timetabling problem of one institution can be vastly different from another. However, it has been generally classified as NP-complete, which favors techniques that employ heuristic and probabilistic search mechanism [19, 20]. Hence, many heuristic algorithms for timetabling have been proposed. They include simulated annealing, constraint logic programming, integer programming and graph coloring heuristic [6, 7, 8, 11, 21, 22, 23]. Heuristic algorithms use the concept of a search space, which contains the complete solution set. Heuristic techniques are employed to explore the search space for the best solution.

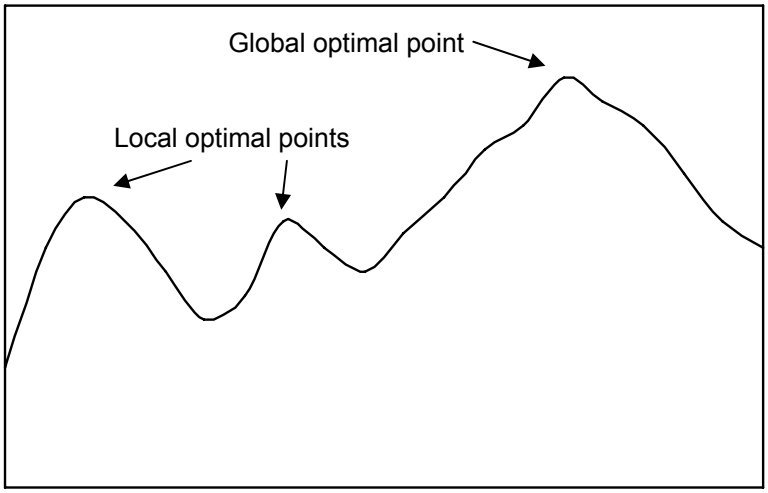



Figure 1. A search space of solutions

The search space is like a landscape of possible solutions as shown in Figure 1. Depending on applications, the fitness of a solution is defined such that solutions that reside at high points in the landscape are considered as optimal solutions. Likewise, solutions that reside at low points are to be avoided as they are poor solutions. Figure 1 shows a search space that contains a global optimal point and two other local points. The key issue in any search technique is to devise an algorithm that is efficient in arriving at a solution that is or near the global optimal point. A secondary issue is that it is capable of escaping from a local optimal point.

Conventional methods of using a mathematical model or some logical rules (eg., integer programming and logic constraint programming) for a problem do not apply very well on university timetabling. One main difficulty is in formulating the problem into a mathematical model. Such

difficulties are due to the characteristics of the classes to be scheduled in a timetable and the interaction among its set of constraints. Classes could be of various duration and they may have to be repeated several times among lecture groups (sub-timetables). This will impose on the order of scheduling of classes with various durations. As the set of constraints are divided into hard and soft constraints, it is possible now to violate some of these soft constraints. Furthermore, some of these constraints may be in the form of interactions among classes rather than just a number representing a certain attribute. The satisfactions of these constraints depend also on the order of which the classes are scheduled. As for some other techniques, such as the simulated annealing, the difficulty is in finding a good single solution.

Hence, researchers are now turning to evolutionary algorithms (EAs), or its hybrid, as a better alternative of solving timetabling problems. The use of EA [9, 10, 24, 25, 26, 27], is a powerful general-purpose optimization technique, which models the principles of evolution [28]. It starts with a set of solutions, which should be distributed across the search space. This set of initial solutions serves as a set of starting points for evolution to take place. Through a series of evolutionary operations, such as crossover and mutation, a globally optimal solution may be obtained.

Many of the research effort on timetabling thus far have placed their emphasis on the examination timetabling problem. However, there is

now an increasing interest in developing EAs for curriculum timetabling [13, 30, 31, 32, 33, 34]. Their solutions tend to be specific to a particular institution and are usually not applicable to other institutions [29]. The main reason is that there is a lack of a uniform definition of the timetabling problem. Another reason is that there are various levels of complexities that are not properly categorized in these solutions, especially in the case of curriculum timetabling. The proposed algorithm developed in this thesis attempts to alleviate such short comings.

A considerable amount of material in this thesis will be dedicated to curriculum timetabling, Though it may look similar to examination timetabling, it has a higher level of complexity. However, the two types of timetables do share many common features and a uniform framework can be presented.

## 1.5. Preview of Thesis

This thesis first examines the characteristics of the timetabling problem of a university. This is then followed by the definition of the problem. With the development of the formulation, a novel EA, which is named as the multi-stage co-evolutionary algorithm (MSCOA), will be presented.

This thesis approaches the timetabling problem in a unified way. It starts by grouping the different types of timetables generally used by various types of higher-level educational institutions. The common platform is

the level of complexity for the different types of timetables. This relates to the constraints and the type of classes involved in the planning of the timetable [19, 20, 35].

The problem can be visualized as a problem of allocating different types of items into a set of small containers. The different items can be viewed as the different types of classes and the set of small containers as timeslots, each with a set of available teaching facilities. There are constraints that will dictate whether certain items can be mixed together in a small container. All the items must eventually be allocated. In fact, this is the kind of thought that will pass through the mind of a human planner using a manual method. Eventually all the items must be present in different proportions in the small containers. If we arrange the small containers in a fixed linear order, it looks like a fixed-length permutation of items. The planning of a timetable is then reduced to the problem of searching for the best permutation of items over the set of timeslots. Based on this view, the size of the set of all feasible permutations can be estimated. With this, we can estimate other parameters, such as the probability of success.

Soft constraints or preferences [31, 32, 36] are used to compute the quality or fitness (in evolutionary jargon) of a permutation. A popular measure is one that measures the amount of violations to such constraints for every timetable generated.

In view of the above formulation, an evolutionary approach [18, 21, 22, 30, 37] has been found to be effective in solving a permutation problem. The evolutionary approach [24, 25, 26, 27, 28] starts with a set of feasible solutions, evolves through a set of evolutionary operators and selects the next generation through a fitness function. Hence, the key issues in this thesis are on the development of an evolutionary algorithm. This involves the building of a diverse set of initial solutions, devising a set of operators and formulating an appropriate fitness function.

Most of the works done so far, based on an evolutionary approach, is on finding a solution to an examination timetable [23, 38, 39, 40, 41]. The approach used is very similar to the conventional methods used for other types of problems. This is commonly termed as a genetic algorithm approach, which uses both a crossover and mutation operation in each evolutionary step. However, it has been found that for a fixed length permutation problem such as a timetabling problem, a crossover operation is not necessary and at times even counter-productive. Essentially, in a timetable, a class must be scheduled in only one timeslot. The process of crossover will result in some classes being scheduled in more than one timeslot, while others are not scheduled at all. To resolve this dilemma requires the design and execution of a repair algorithm. There are several works on timetabling that use only the mutation operation [2, 39, 42].

In view of the above, the author proposes a novel mutation technique, coupled with a modified form of selection scheme using elitism, introduced in this thesis. It offers a better chance of finding an optimal solution [43, 44, 45, 46]. There are three aspects in this mutation technique. The first aspect is that a substring is randomly defined over the chromosome. It is like having a small moving window over the entire chromosome. Mutation can only happen in this window. The second aspect is the issue of identifying the right window size. The window size will determine the diversity of the approach, which is a salient feature of an evolutionary approach and the ability to hill-climb an optimal point. It has been found that a window size, that reduces progressively with iterations give a better solution. There are many possible functions that can achieve this goal. The third aspect relates to the fact that a mutation may not necessarily bring about a better solution. However, it is also not advisable to discard every mutated chromosome that is inferior. Very often, an inferior mutated chromosome may, at times, through several mutations, produce an even better chromosome. In a search problem, this is a recognized problem of finding an escape path from being trapped in a local optimal. In view of these, an elite form of evolution is maintained at the end of a mutation of a population of chromosomes. This is akin to keeping a copy of the best solution across all generations.

A second major difference is to introduce the concepts of co-evolution [43, 47] and multi-stage evolution [39, 44, 48, 49] to the overall algorithm. These are used as added meta-heuristics to a conventional

evolutionary approach. Incorporating meta-heuristics is gaining popularity in recent works [1, 40, 50, 51]. A timetable is closely related to the curriculum structure, the size of the students and other intrinsic requirements. These introduce different levels of complexities in the form of interactions among the different types of classes and student groups. This is akin to having sub-species competing or cooperating for the same set of resources [28, 34, 43, 47]. The problem is especially acute in the early stage of building an initial population of solutions. It is much easier to first identify sub-species and build each of them separately. The sub-species are ranked according to their constraining effect upon others. Instead of competing, a modified form of cooperation is used among the sub-species. This is in the form of allowing the more constrained species to evolve first, followed progressively by lesser constrained sub-species. This is viewed as a multi-stage approach. However, earlier species may have to be reallocated if the current species has difficulty in its evolution. This requires cooperation among the sub-species. A hybrid combination of co-evolution and multi-stage evolution is then necessary.

The application timetables are taken from the undergraduate courses of the School of Electrical and Electronic Engineering (EEE) in Nanyang Technological University (NTU) [14]. Due to a large number of students and subjects, scheduling these timetables is very complex, rendering the existing manual techniques ineffective. As such, a considerable amount of effort is put into examining the current intrinsic structure and the

problems arising thereof. Three case studies, based on the examination timetable, curriculum timetable and the subject allocation system respectively, will be built to ascertain the performance of the proposed evolutionary approach.

A novel EA approach is necessary as it has been found that most of the existing EA approaches were applied solely to examination timetabling. And those approaches that do cater to curriculum timetabling cannot be applied directly to the EEE curriculum timetabling problem.

There have been several attempts in the past to purchase commercial timetabling software to solve EEE timetabling problems. Several difficulties were encountered. Most of these commercial products cater for small class size and uniform class duration. They do not provide options for planning different number of lecture groups taking the same class. As such, a lot of tweaking is necessary. Furthermore, as the curriculum in NTU is still evolving, it is necessary to change the heuristics rules frequently. Without the source codes, EEE has to rely on suppliers to update these changes. This may slow down the update process and incur extra costs.

# Chapter 2 Timetabling in NTU

This chapter describes the existing timetabling system used in the College of Engineering (CoE) of Nanyang Technological University (NTU) [14]. The information described in this chapter is based on the Academic Year 2002/03. It discusses the need to automate the various timetable scheduling at NTU and explains how the present timetabling system is evolved and why the University has chosen to develop the timetable software internally. The current system employs a hybrid combination of general computing tools (such as the Microsoft Office tools), web-based programming paradigms and special optimization techniques. Some general design and implementation details of the software are presented as part of the evolving CoE timetabling system. It also shows that the most suitable techniques are those optimization techniques that employ stochastic or probabilistic search, such as that of an evolutionary approach.

## 2.1. Terminology

This introductory section defines terms that are used in the NTU timetabling system. These terms are typed in *italics* below.

A *school* or *department* within a university offers at least a *course* or program leading to a degree. Generally, a student may need to study 3

to 4 years to complete a *course*. For every *year of study*, he is required to read a number of *subjects*. An *academic year* is usually divided into 2 or 3 *semesters*, with breaks of 1 to 2 months. In every semester, a number of *subjects* are offered to every *year of study*. Depending on the demand, some of these *subjects* may be offered in both *semesters*.

A *subject* is conducted in 2 or 3 classes per week. There are several types of classes. Typically, in the context of EEE, a *theory subject* is conducted in 2 *lecture classes* and 1 *tutorial class*, while a *practical subject* is conducted in either 1 *laboratory* or *design* or *project class*. Hence, there are 5 types of classes.

## 2.2. Overview

The College of Engineering at NTU comprises numerous engineering schools such as the Civil, Computer, Electrical, Materials and Mechanical Engineering School. The college has a total of about 10,000 undergraduate students, 3,000 postgraduate students and 600 academic staff. Every engineering student needs to read subjects from a common pool of subjects (broad-based curriculum) as well as from a pool of school-specific subjects (school curriculum). This has made the design of curriculum and examination timetables extremely complex and tedious. The timetable of each engineering school cannot be planned independently without consideration of the timetables of other engineering schools. Common subjects require large lecture theatres

(LTs) and laboratories (Labs). There is a need for an innovative timetabling sequence to schedule lectures, tutorials and laboratories for about 3,800 students.

The curriculum structure has put pressure on:

- intelligent use of the LT resource;
- optimization techniques to minimize number of times students' need to change LTs and the time it takes them to walk between lectures;
- lab cycling sequence design that can cater to all the lab modules for the various lab groups;
- Examination timetables that can schedule all the common subjects without adverse effect on the school-specific subjects.

Timetabling in the CoE is an assembly of several sub-systems modules. As mentioned earlier, these modules are designed by exploiting web-based computing and special scheduling techniques. The university has to integrate the various software modules of curriculum scheduling to facilitate resource planning, student timetable planning, subjects' registration procedure and scheduling and distribution of staff workload.

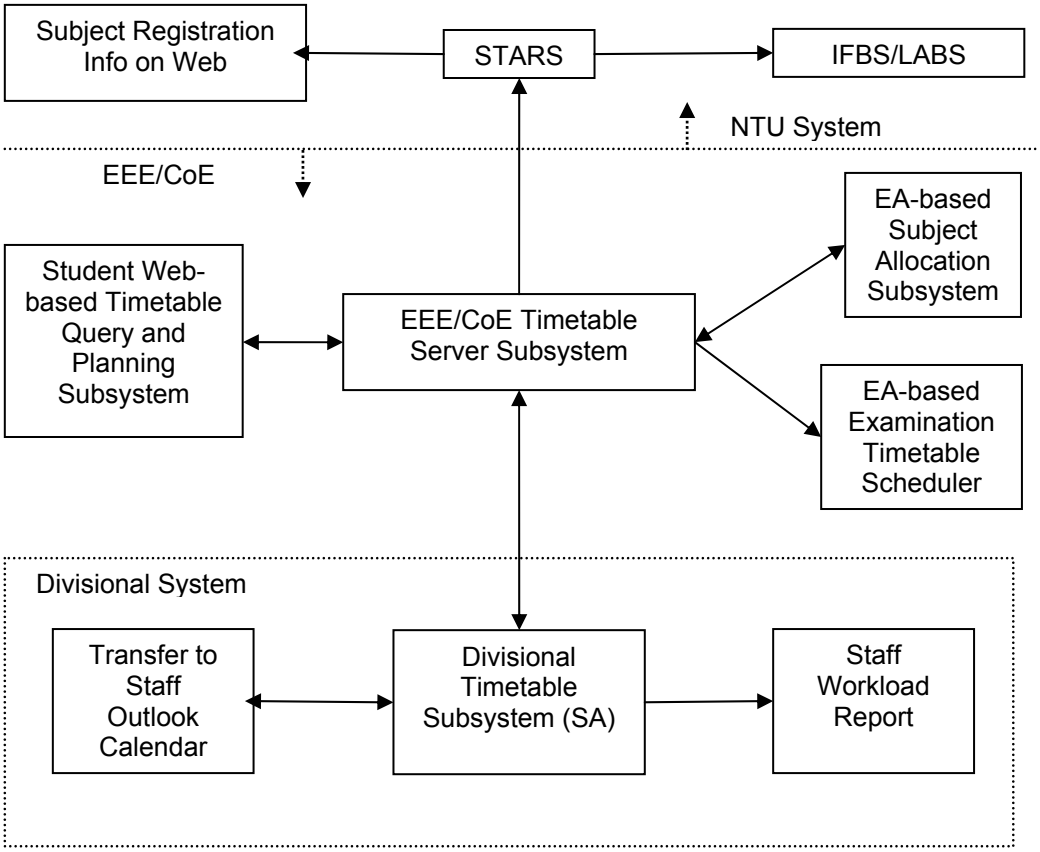



Figure 2. CoE Timetabling System

Figure 2 shows the interface diagram of the various sub-systems of the CoE timetable system at NTU. The system consists of three-tier sub-systems.

- The STudent Automated Registration System (STARS) in the university tier.
- The EEE/CoE Timetable Server Sub-System in the school tier.
- The Divisional Timetable Sub-System in the divisional tier.

The sub-systems work not only towards the functional needs in their respective tier but also exchange data in a coordinated manner for the proper operation of the entire CoE Timetable System.

In the university tier, STARS is the main system where all the schools upload their subject and timetable information to populate its database. The information gathered from all the schools are then converted and published as static html pages for access by students during subject registration.

STARS interfaces with the Interactive Facility Booking System (IFBS) and the Lab Facility Booking System (LABS). The former controls all the bookings of lecture theatres and tutorial rooms while the latter controls the bookings of communication skills and engineering labs. Unlike IFBS, the bookings of lab facilities are not open to all academic staffs. LABS was designed specifically for the sole use of curriculum teaching so that lab facilities can be booked automatically, whenever their schedules are populated into the STARS database.

In the school tier, the EEE/CoE Timetable Server is responsible for generating all the curriculum timetables for transfer to STARS. At the moment, due to the complexity of the Engineering program, the curriculum timetables are manually designed. However, plans are in place to replace the manual approach by an EA-based technique. This technique will be further discussed in details later in this thesis. Once the design of the curriculum timetable is complete, the database that stores all the curriculum information is also created. Based on the information in the database, the detailed teaching assignment form for each subject

in the curriculum timetable can be generated. In EEE, the teaching assignment form shows the days and hours when the lectures/tutorials are conducted for each of the thirteen weeks of a semester. To complete the form, the divisional timetable coordinators work with the Heads of Divisions to assign staffs for teaching duties according to their specialization and capabilities. The teaching assignment form for practical subjects reflects how the lab modules will be scheduled in a certain cycling sequence throughout the semester. The scheduling of lab modules is performed via a simple AI algorithm to optimize limited lab resources. Lab makeup classes are also scheduled when a holiday occurs. Samples of teaching assignment forms for all subjects can also be viewed at http://timetable.eee.ntu.edu.sg/tt/ .

Based on the curriculum timetables, examination timetables can also be generated. An EA-based Examination Timetable Scheduler has been designed that uses the curriculum information as inputs to optimize its exam schedule. Subjects that are taught in the same time-slot of the curriculum timetables can be scheduled in the same examination timeslot. Other information such as the number of students in each subject can be used to assign the appropriate exam hall with the necessary seating capacity.

The Web-Based Query and Timetable Planning Sub-System allows students to examine the teaching assignment form for each subject and obtain information concerning the lecturers and tutors.

Through these dynamic html pages, students make their class selections and create their own timetables. The timetable planner software tool will then map their chosen subjects to respective indices so that these indices can be entered into STARS during subject registration. Plans are also in place to allow students to transfer their pre-planned timetables from the web to the EA-Based Subject Allocation Sub-System automatically.

In the divisional tier, the Divisional Timetable Sub-System runs the Staff Workload Scheduling software to schedule teaching staff to meet the teaching duties of the division. The software uses a simulated annealing algorithm to optimize equal distribution of staff workload. Users can also choose to manually assign the staff workload so that they can have full control over the time slots assigned. The assignment is said to be complete when all teaching loads within the division are assigned satisfactorily.

## 2.3. Motivation for CoE Timetabling System Project

The University has always regarded timetable planning and scheduling as one of the important components of curriculum teaching. As the university's student intake and the number of subjects offered continue to increase, scheduling of curriculum and examination timetables becomes increasingly complex and tedious. In 1981 when NTU was

established, pencils and papers were sufficient to plan curriculum timetables since the enrolment was less than a few hundred students. Today NTU has more than 23,000 students and over 1,000 academic staff.

With the advancement of information technology, commercial software is readily available to solve many scheduling problems. Commercial software packages make use of the state-of-the-art algorithm such as genetic algorithm and simulated annealing. These are able to solve general scheduling problems. However, they are not able to dedicate to unique curriculum changes and future IT innovations without incurring much cost. NTU is a relatively new university and is still expanding. Therefore, changes to its curriculum can be expected. It would be too costly to constantly engage a software supplier to upgrade the software package whenever there are curriculum changes and new IT innovations. In addition, commercial software packages are usually sold and distributed in executable codes rather than source codes, to protect future products sale. This makes software maintenance and integration with existing in-house systems such as STARS and IFBS extremely difficult. Specific university needs such as labs and makeup classes in lieu of public holidays and cancelled classes are usually not taken care of in most commercial software packages. Some vendors are unwilling to code these features as many universities do not require these features and even if they are willing to code, the cost incurred will be substantial. For a university such as NTU which has already developed

a portion of timetable software over the years, it is not feasible to go to vendors which do not supply their source code. System integration between existing and vendor software will be difficult and much costs will be incurred to hire new vendors to do the interface works. The University evaluated several commercial software packages and found them lacking in the features unique to NTU.

Thus, there is a need to develop an automated scheduling system that meets EEE/CoE timetable requirement so that the software can be upgraded internally as and when the need arises.

## 2.4. Examination Timetable

A memetic genetic algorithm [2, 52, 53] is used for the examination timetable for school of EEE. The memetic algorithm is a hybrid of the heuristic sequencing and evolutionary method that can outperform either method on its own. The memetic algorithms are able to find a similar solution to the search algorithms in a similar length of time. The advantage of the memetic algorithm using hill climbing as the local optimizer is that it has the capability to continue and find a better solution in a longer amount of time. It has a head start over genetic algorithm by seeding the initial population with feasible and locally optimum timetable. The search is then guided towards the best sector of the solution space and hence the best timetable.

## 2.5. Curriculum Timetable

The timetables used in NTU are planned for an academic semester, which is 13 weeks long. There are two semesters in a year. Of all the schools in NTU, EEE is the largest. Students join EEE in their second year, after successfully completing a common-curriculum in the first year. Third year students are attached to the industry for practical training in semester two. As such, there is no scheduled teaching for third year students in semester two. Final year subjects are divided into six option groups and each group has its own prescribed subjects. In addition, final year students have to study a number of free elective subjects which are of a general nature.

All the students have to study core subjects pertaining to their current year of study. Most of these core subjects are offered in both semesters. Furthermore, students who have failed in some subjects have to retake and pass them within the university's specified time frame.

### 2.5.1. Subjects and Classes

There are two types of subjects: theory and practical. There are five types of classes: lecture, tutorial, project, laboratory and design. They differ in their duration and frequency per week. The lecture and tutorial classes are of one-hour duration. A theory subject is usually conducted in a number of lecture classes and a tutorial class. Hence, these classes

are also termed as theory classes. The other three classes are termed as practical classes as they require the use of laboratory equipments and are of two- or three- hour duration.

A student should attend all the subjects associated to a year of study. Most of the lecture classes are conducted in lecture theatres (250 to 350 seats), while tutorial classes are conducted in tutorial rooms (about 30 seats). The number of students studying a theory subject is usually very large, and can be as large as 900. Hence, for every theory subject, a greater number of tutorial classes than lecture classes are required. However, a student needs to attend only one of the tutorial classes (associated with a theory subject) in a week.

In addition to the theory subjects, students are required to read practical subjects too. Practical classes are also small in size (about 30). In terms of venues, they do not compete with other classes that engage lecture theatres and tutorial rooms. However, they do compete with other types of classes over the limited number of timeslots.

### 2.5.2. Student Groupings

Table 1 shows the breakdown of the second and third year students into smaller groups. Students of a year of study are divided into three lecture groups. Students belonging to a lecture group should attend lecture classes designed for that group. Each of this lecture group is further

divided into smaller subgroups for tutorial classes, laboratory classes, design classes and project classes. Classes that are taught in one lecture group must be repeated for the other lecture groups in the same year of study.

Table 1. Student groupings of second and third year students

| Year of Study | Second Year 3 lecture groups | | | Third Year 3 lecture groups | | |
|---|---|---|---|---|---|---|
| Lecture group (gp) | SA gp | SB gp | SC gp | TA gp | TB gp | TC gp |
| Tutorial subgroup (subgps) | TS01-TS12 12 subgps | TS13 - TS26 14 subgps | TS27 - TS38 12 subgps | TT00 - TT13 13 subgps | TT14 - TT26 13 subgps | TT27 - TT38 12 subgps |
| Laboratory subgroup (subgps) | LS01 - LS12 12 subgps | LS13 - LS26 14 subgps | LS27 - LS38 12 subgps | LT01 - LT13 13 subgps | LT14 - LT26 13 subgps | LT27 - LT38 12 subgps |
| Project subgroup (subgps) | PS01-PS12 12 subgps | PS13-PS26 14 subgps | PS27-PS38 12 subgps | PT01-PT13 13 subgps | PT14-PT26 13 subgps | PT27 - PT38 12 subgps |
| Design subgroup (subgps) | DS01-DS12 12 subgps | DS13-DS26 14 subgps | DS27-DS38 12 subgps | DT01-DT13 13 subgps | DT14-DT26 13 subgps | DT27 - DT38 12 subgps |

### 2.5.3. Duration and Frequency of Classes

Table 2 shows the duration and frequency of the different types of classes. Frequency is the number of times a class is conducted per week. For example, a student in a lecture group, reading a theory subject may have to attend 2 lecture classes and 1 tutorial class in a week. At the same time, for practical subjects, the student needs to

attend a laboratory class, a project class and a design class planned for that lecture group.

Table 2. Duration and frequency of classes

| Type of class | Second year | | Third year | | Final year | |
|---|---|---|---|---|---|---|
| | Duration (hour) | Frequency | Duration (hour) | Frequency | Duration (hour) | Frequency |
| Lecture | 1 | 2 or 3 | 1 | 2 or 3 | 1 | 2 or 3 |
| Tutorial | 1 | 1 | 1 | 1 | 1 | 1 |
| Laboratory | 3 | 1 | 3 | 1 | NA | NA |
| Project | 3 | 1 | 3 | 1 | 10 | 1 |
| Design | 2 | 1 | 3 | 1 | 3 | 1 |

## 2.5.4. Hard and Soft Constraints

Table 3 shows a partial timetable used by the second year students. It shows the morning schedule of Monday and Tuesday. This is designed for the SA lecture group. Class starts in the morning at 0830.

Table 3. Year 2 Semester 1 partial Timetable for SA – morning classes

<table>
<tr><th></th><th>0830 - 0930</th><th>0930 - 1030</th><th>1030 - 1130</th><th>1130 - 1230</th><th>1230 - 1330</th></tr>
<tr><td>MON</td><td>Lecture<br>E202<br>(LT23)</td><td>Lecture<br>E201<br>(LT23)</td><td>Lecture<br>E203<br>(LT23)</td><td></td><td>Tutorial<br>E201-TS01<br>(TR115)</td></tr>
<tr><td rowspan="3">TUE</td><td>Lecture<br>E120 - L1<br>(LT27)</td><td colspan="3">E221 (LAB)<br>LS05, LS06, LS07, LS08</td><td rowspan="3"></td></tr>
<tr><td rowspan="2">Tutorial<br>E201-TS02<br>(TR114)</td><td>Tutorial<br>E120-TS01<br>(CSkL8)<br>E202-TS02<br>(TR114)</td><td>Tutorial<br>E120-TS01 (CSkL8)<br>E203-TS03 (TR114)</td><td rowspan="2">Tutorial<br>E204-TS03<br>(TR114)</td></tr>
<tr><td colspan="2">E227 (DESIGN)<br>DS03(TR115), DS04(TR116),<br>DS05(TR117),DS06(TR118)</td></tr>
</table>

The first three classes of Monday morning are lecture classes. The entire lecture group attends these lecture classes. As such, no other classes for SA can be placed parallel with them in these timeslots. This consideration constitutes one of the hard constraints. A violation of this constraint will result in an infeasible timetable, as the same group of students cannot be in two different classes at the same time.

The lecturer conducting a lecture class to one lecture group is also conducting the same lecture class to all the other lecture groups. Hence, the same lecture class cannot be planned for the SB and SC lecture group in the same timeslot. In this case, the lecture class, E202, cannot be placed at 0830, Monday for SB and SC. This is another hard constraint, as a lecturer cannot be in more than one place at the same timeslot.

All of these lecture classes are conducted in the lecture theatre, LT23. They are placed consecutively one after another. Having consecutive classes is convenient for the students, allowing them to maximize their time. Since these are lecture classes attended by about 400 students, it is also advisable to have the same lecture theatre for all the consecutive classes. This will minimize the transition time between lectures. This constitutes two of the first soft constraints that are used in the computation of the fitness function.

There is a free timeslot at 1130 on Monday for the SA lecture group. The students could use this free timeslot for their lunch. In EEE, lunch is scheduled between 11:30am to 1:30pm. However, there are instances when students may have to take their lunch before or after this lunch period. This is considered undesirable and though it is not considered to be a violation of the hard constraint, it constitutes another soft constraint.

The next class is a tutorial class. This tutorial class is planned for TS01 tutorial subgroup. The duration for a tutorial class is one hour. Tutorial classes belonging to SA cannot be placed parallel with lecture classes planned in this timetable. For every lecture class, there may be 12 tutorial classes related to it. Since, there are many tutors available in conducting the tutorial classes, parallel tutorial classes can be placed in the same timeslot without violating any hard constraints.

Tuesday morning starts with two parallel classes, a lecture class and a tutorial class, even though it was mentioned earlier that a lecture class could not be placed in parallel with a tutorial class. Not every student in this lecture group is required to read the lecture class, E120. As such, these students can attend the tutorial class.

Tuesday 0930 timeslot has three parallel classes, a laboratory, tutorial and a design class, each of a different duration. There is only one laboratory subject for each lecture group. Due to a limited number of

laboratory resources, there should not be more than four laboratory classes for each lecture group at any timeslot. This constitutes another hard constraint. This constraint applies to the design classes too.

Table 4. Year 2 Semester 1 partial Timetable for SA – afternoon classes

<table>
<tr><td></td><td>1230 - 1330</td><td>1330 - 1430</td><td>1430 - 1530</td><td>1530 - 1630</td><td>1630 - 1730</td></tr>
<tr><td rowspan="3">MON</td><td rowspan="3">Tutorial<br>E201-TS01<br>(TR115)</td><td colspan="2">E221 (LAB) : LS01, LS02, LS03, LS04</td><td rowspan="3">Tutorial<br>E203-TS02<br>(TR119)<br>E204-TS01<br>(TR120)<br>E204-TS02<br>(TR115)</td><td rowspan="3">Lecture<br>GE02<br>(LT28)</td></tr>
<tr><td colspan="2">E227 (DESIGN)* : DS01(TR119), DS02(TR120)</td></tr>
<tr><td>Tutorial<br>E202-TS01<br>(TR115)</td><td>Tutorial<br>E203-TS01 (TR115)</td></tr>
<tr><td>TUE</td><td></td><td>Lecture<br>E202<br>(LT23)</td><td>Lecture<br>E204<br>(LT23)</td><td>Lecture<br>E203<br>(LT23)</td><td></td></tr>
</table>

Table 4 shows the afternoon schedule of the Monday and Tuesday timetable. There are three parallel classes planned in the Monday 1330 to1430 timeslot, each of a different duration. There are four laboratory classes, one tutorial class and two design classes. There is a break between the morning and afternoon laboratory classes. The laboratory needs at least an hour break to prepare for the next laboratory class. This is a hard constraint that applies to all practical classes.

There should not be more than one lecture class of a theory subject on the same day for the same lecture group. This is another hard constraint.

### 2.5.5. Developing a Novel Evolutionary Approach

The structure of the EEE curriculum timetable renders a conventional evolutionary algorithm approach ineffective. An evolutionary approach augmented with some form of heuristics has to be explored. This requires detailed definition and formulation of the scheduling complexities of a timetable. In the formulation, the numbers of all the solutions, feasible solutions and infeasible solutions are computed. Based on these numbers, the probability of success is computed to give us a guide to the success and efficiency of a particular technique in finding a solution. This results in arriving at an evolutionary approach that incorporate meta-heuristics, such as multi-stage and co-evolution.

## 2.6. Subject Allocation System

A Web-Based Timetable Planning Sub-System has been developed to assist students in registering their subjects online. For every subject, students must select the classes that are associated with it. Once the selection is confirmed, the student timetable will be created accordingly as shown in Figure 3. As the student adds other subject classes to the timetable, clashes between classes will be checked. Each class comes with a subject index, so classes that do not clash will be listed along with their subject indices at the bottom of the timetable. As the system is an on-line system, the number of vacancy in each class will be deducted immediately. If a class is full, students will be advised immediately and they can then try another class until all their subjects are registered.

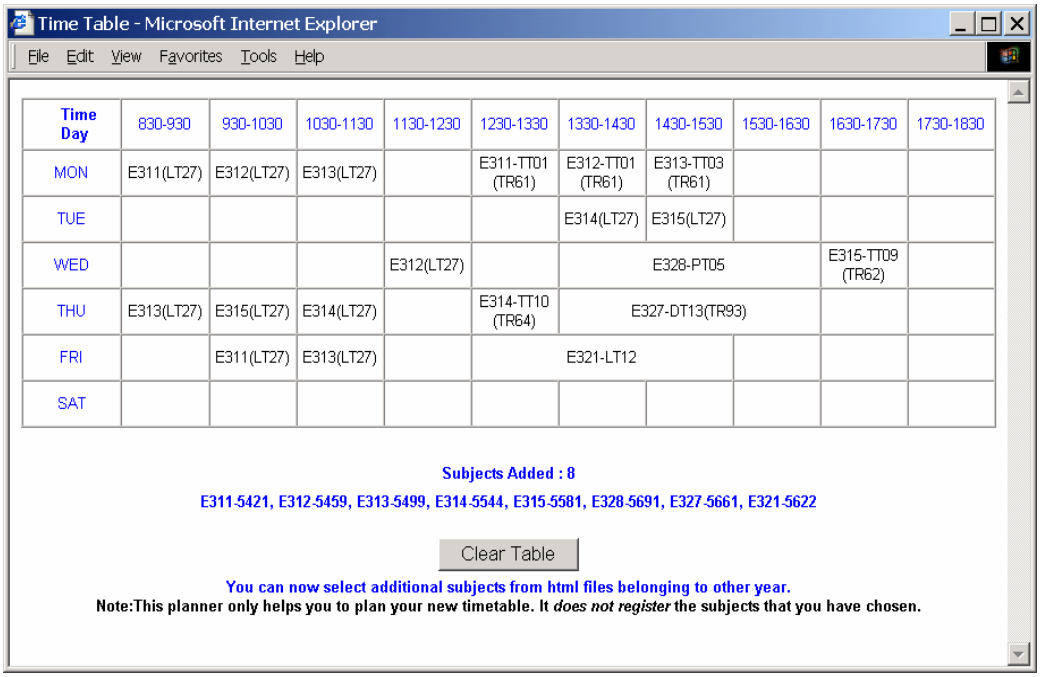



Figure 3. Web-based Timetable Planning Subsystem

The current on-line registration system is unable to cope when a few hundred students attempt to register their subjects simultaneously. Hence, the system administrator divides the students into several smaller groups. Each group has up to two hours to register their subjects, but in reality, students compete fiercely within the first two minutes for the best timeslots. Students who are able to key in their entries speedily stand a better chance of getting their desired timeslots. Unnecessary anxiety and stress are built up among the students during subject registration days as many fear that they would be unable to get into their preferred timeslots.

An off-line Subject Allocation Sub-System, which is based on the novel evolutionary algorithm, has been developed [45]. It allows students up to several days to submit their chosen subjects. The Sub-System would only be run after the submission is closed. Therefore, students can take their time to enter their choices and this alleviates the students' anxieties and the need for them to rush to key in their choices. It is able to increase the hit rate of the number of students with timetables of their choices. In the curriculum timetable planning, with its limited resources (lecture theatres, tutorial rooms and laboratories), it is not possible to satisfy all students with their preferred classes. Some students will lose out during the subject allocation process. For these students, the sub-system will fall back on the second best timetable that they had selected. Of course, the sub-system can continue to fall back on the next less favorable choice. Results have indicated that it is possible to achieve good success rate with merely two choices of timetables entered by the students. Hence, it has been decided that two choices are all that the subsystem would require in order not to bore the students with too many timetable plans.

## 2.7. Staff Allocation System

The Staff Allocation System was the earliest allocation system developed in EEE. It is based on the simulated annealing optimization technique. Teaching assignment forms for the various subjects will be generated when the curriculum timetable is completed. Table 1 shows a

blank partial teaching assignment form for a particular subject. The system ensures that every teaching staff has an even proportion of teaching assignments without any clashes of timeslot. This system is already fully operational and has been found to be effective and efficient. As such, it is not used in testing the novel evolutionary algorithm proposed in this thesis.

Table 5. Partial Teaching Assignment Form

| WEEK | DATE | TUE<br>1330 – 1430<br>LT25 | THUR<br>1030 – 1130<br>LT25 |
|---|---|---|---|
| 1 | 26 July - 31 July 2004 | | |
| 2 | 02 Aug - 07 Aug 2004 | | |
| 3 | 09 Aug - 14 Aug 2004 | | |
| 4 | 16 Aug - 21 Aug 2004 | | |
| 5 | 23 Aug - 28 Aug 2004 | | |
| 6 | 30 Aug - 04 Sep 2004 | | |
| 7 | 06 Sep - 11 Sep 2004 | | |
| ... | … | … | … |

# Chapter 3 Defining the Timetabling Problem Statement

This chapter examines ways in which a timetabling problem could be defined and the best approach to solve it. The EEE curriculum timetable is used for the discussion.

## 3.1. Teaching Resources and Activities

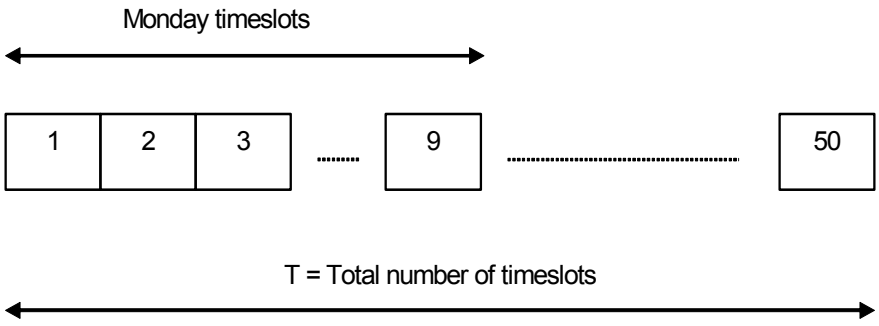



Figure 4. Timeslots over a week

Teaching activities are planned over a week. A week is divided into timeslots. Each timeslot is of a fixed duration (example, one hour). Total number of timeslots for a week is an important resource. Timeslots can also be grouped together, such as according to the day of the week. So there is a group of timeslots for Monday, another for Tuesday and so on.

There are 9 timeslots on each day, Monday to Friday and 5 timeslots on Saturday morning. This gives a total of 50 timeslots in one week as shown in Figure 4.

There are five types of classes, namely lecture, tutorial, laboratory, project and design to be scheduled in a timetable. The lecture and tutorial class are usually 1-hour in duration. The design class is of 2-hour duration or 3-hour duration. The rest are the project and laboratory class, which are 3 hours in duration. Hence, it is also possible to classify classes according to their duration.

Very often, there is no lecture theatre big enough to accommodate a large group of students reading a lecture class. The students have to be divided into smaller groups. Each of these groups is termed as a lecture group. Students in each lecture group are required to take the same set of lecture, tutorial, laboratory, design and project classes as those in other lecture groups for the same year of study. Typically, there are three lecture groups per year of study. The number of lecture groups per lecture class introduces some level of complexity which will be discussed later.

## 3.2. Timetabling Problem is a Permutation Problem

A timetabling problem can be visualized as a problem of allocating a collection of items into a fixed set of containers as shown in Figure 5.

The items in the case of a curriculum timetable are classes that need to be scheduled. A container is a teaching facility, such as lecture theatre or tutorial room or laboratory, found available in a timeslot. It is possible to have several teaching facilities available in one timeslot. Each timeslot is of equal duration with another. The number of timeslots is fixed.

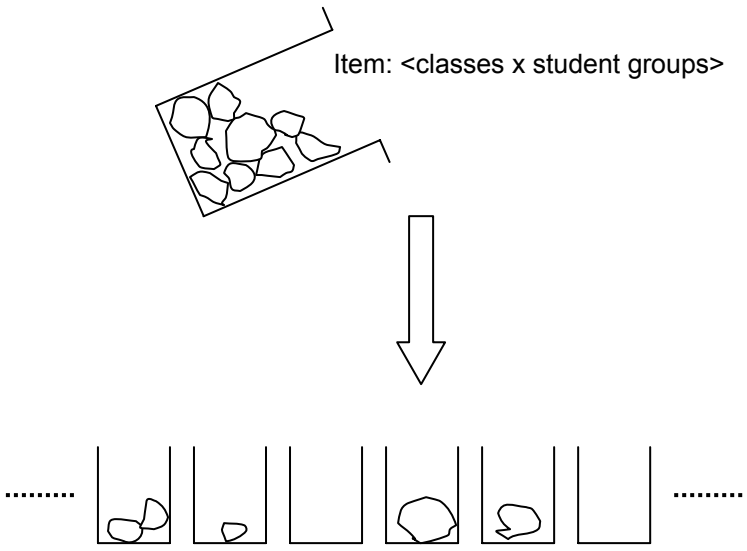



Figure 5. Timetabling as an allocation problem

The objective is to allocate completely all the items into the containers subject to a set of constraints. There is usually more than one way to do this. If the containers are arranged in a fixed linear order, an allocation may be viewed as a permutation of fixed items as shown in Figure 6. Note that it is possible to have containers that are not allocated with any

item. A permutation is considered feasible if it satisfies all the hard constraints. It is more superior to another if it satisfies more of the soft constraints.

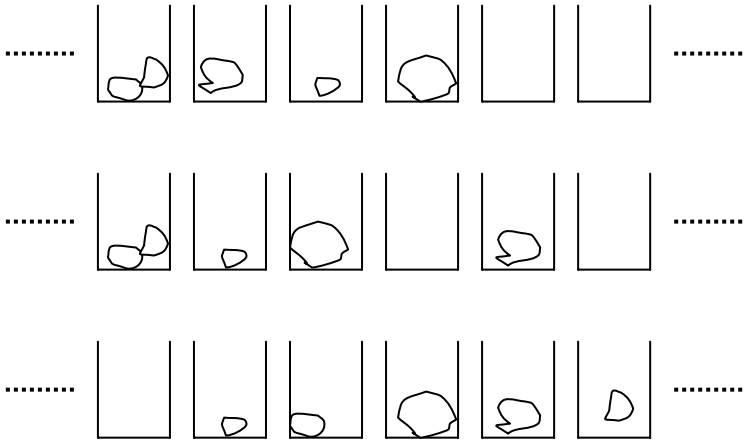


Figure 6. Timetabling as a permutation problem

## 3.3. Slack Ratio (*SR*)

Let *T* be the total number of timeslots, *R* be the required number of timeslots. *SR* is defined as (*T*-*R*)/*T*. In order to obtain at least one feasible solution, *SR* must be greater or equal than 0($SR \geq 0$).

The following sections examine the factors that affect *SR* and consequently the set of feasible solutions or the feasible solution space.

## 3.4. Effect of the Number of Lecture Groups on *SR*

Assume that a student in a lecture group reads $m$ lecture classes. Each class is of identical duration $d = 1$. Assume also that the lecture classes are not allowed to clash in the same timeslot. To simplify the discussion, there is a lecture theatre for a lecture class in each timeslot.

Case 1: One lecture group

$R = m\ d$

$SR = (T - m\ d) / T = 1 - m\ \ d / T$

$SR \geq 0$ if $m\ d <= T$

Case 2: Two lecture groups

$R = 2\ m\ d$

$SR = (T - 2\ m\ d) / T = 1 - 2\ m\ d / T$

$SR \geq 0$ if $2\ m\ d <= T$

Case 3: $g$ lecture groups

$R = g\ m\ d$

$SR = (T - g\ m\ d) / T = 1 - g\ m\ d / T$

$SR \geq 0$ if $g\ m\ d <= T$

Note

- If $g\ m\ d = 0$ then $SR = 1$. This situation cannot happen since it means that there is no class to schedule.

- If $g\ m\ d = 0.5\ T$ then $SR = 0.5$. There is more than enough timeslots for scheduling.
- If $g\ m\ d = T$ then $SR = 0$. This implies that there will be no extra timeslot. This is a very tight situation.
- If $g\ m\ d > T$ then $SR < 0$. Some classes are not allocated a timeslot. No solution is then possible.

*SR* is an indicator of the ease of finding a solution. It is easier to find a solution when $SR = 0.5$ than when $SR = 0$. As *SR* approaches 0, the search for a solution becomes progressively more difficulty. It is not possible to find a solution for cases where $SR < 0$. In such cases, a possible way out is to reduce the number of lecture groups by optimizing the usage of larger size lecture theatre.

## 3.5. Fragmentation

Assume that there are two types of classes. The number of classes for one type is $m_1$ and the other is $m_2$, with duration $d_1$ and $d_2$ respectively, $d_1 < d_2$.

Case 1: One lecture group

$$R = m_1 d_1 + m_2 d_2$$

$$SR = (T - (m_1 d_1 + m_2 d_2)) / T = 1 - (m_1 d_1 + m_2 d_2) / T$$

$$SR \geq 0 \text{ if } (m_1 d_1 + m_2 d_2) <= T$$

Case 2: Two lecture groups

$R = 2\ (m_1\ d_1 + m_2\ d_2)$

$SR = (T - 2\ (m_1\ d_1 + m_2\ d_2)) / T = 1 - 2\ (m_1\ d_1 + m_2\ d_2) / T;$

$SR \geq 0$ if $2\ (m_1\ d_1 + m_2\ d_2) <= T$

Case 3: *g* lecture groups

$R = g\ (m_1\ d_1 + m_2\ d_2)$

$SR = (T - g\ (m_1\ d_1 + m_2\ d_2)) / T = 1 - g\ (m_1\ d_1 + m_2\ d_2) / T;$

$SR \geq 0$ if $g\ (m_1\ d_1 + m_2\ d_2) <= T$

Though the effect of the number of lecture groups on *SR* is similar to the earlier section, another factor has come into effect which makes allocation difficult. When there are at least two classes that are longer than 1 timeslot, there arises the possibility of producing fragmentation in the timetable.

Assume that duration $d = 3$ timeslots for class $C_1$, $C_2$, $C_3$

*allocation* 1: no fragmentation

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $C_1$ | $C_1$ | $C_1$ | $C_2$ | $C_2$ | $C_2$ | $C_3$ | $C_3$ | $C_3$ |

*allocation* 2: fragmentation occurs

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $C_1$ | $C_1$ | $C_1$ | | $C_2$ | $C_2$ | $C_2$ | | |

Figure 7. Presence of fragments reduces the feasible solution space

Figure 7 shows two allocations of class $C_1$, $C_2$ and $C_3$ over a timetable of nine timeslots. As each class is of 3 hour duration, three consecutive timeslots are needed. Fragmentation occurs when there is a number of consecutive empty timeslots that are not able to accommodate any outstanding classes with longer duration. As shown in Figure 7, in the case of *allocation* 2, there is one empty timeslot between $C_1$ and $C_2$ and two consecutive empty timeslots after $C_2$. It is now not possible to allocate $C_3$. The *SR* is the same for both allocations, yet *allocation 2* is not a feasible solution.

## 3.6. Fragmentation and the Feasible Solution Space

The following illustrations discuss in detail the effect of class duration on the possibility of producing fragmentation in an allocation and thereby affecting the feasible solution space.

Let us start with an assumption that a timetable consists only 4 1-hour timeslots ($N$ = 4). We will allocate classes with different duration for each of the following cases. The number of classes is fixed such that *SR* >= 0. In each of the cases, we assume that the duration of each class is the same. Let us also define the probability [54] of obtaining a feasible solution in a single iteration as:

*PFS* = number of feasible solutions / total number of solutions

Case 1 ($d$ = 1)

Assume that there are 4 classes ($C_1$, $C_2$, $C_3$, $C_4$), each of 1 timeslot in duration ($d$ = 1), to be allocated.

$SR$ = (4 - 4) / 4 = 0

Number of possible allocations = 4! = 24 possible feasible solutions.

Number of infeasible solutions (or partial solutions) = 0

Total number of solutions = 24

$PFS$ = 24 / 24 = 1

Case 2 ($d$ = 2)

Assume that there are 2 classes ($C_1$, $C_2$) each of 2 timeslots in durations ($d$ = 2) to be allocated.

$SR$ = (4 - 4) / 4 = 0

As seen in Figure 8,

Number of possible allocations = 2! = 2 possible feasible solutions.

Number of infeasible solutions due to fragmentation = 2

Total number of solutions = 2 + 2 = 4

$PFS$ = 2 / 4 = 0.5

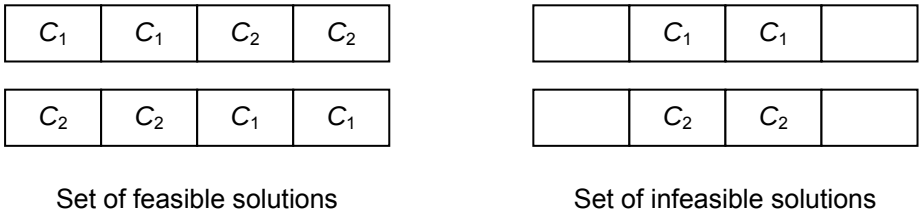



Figure 8. Allocating 2 classes of 2 timeslots duration into 4 available timeslots

Case 3 ($d$ = 3)

Assume that there is 1 class ($C_1$), which is 3 timeslots in duration ($d$ = 3), to be allocated.

$SR$ = (4 - 3) / 4 = 0.25

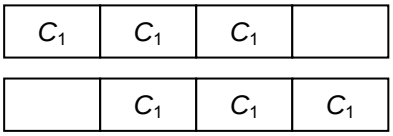
As seen in Figure 9,

Number of possible allocations = 2 possible feasible solutions.

Number of infeasible solutions due to fragmentation = 0

Total number of solutions = 2

$PFS$ = 2 / 2 = 1




Set of feasible solutions

Figure 9. Allocating 1 class of 3 timeslots duration into 4 available timeslots

Case 4 ($d$ = 4)

Assume that there is 1 class ($C_1$), which is 4 timeslots in duration ($d$ = 4), to be allocated.

$SR$ = 4 / 4 = 1

Number of possible allocations = 1 feasible solution.

Number of infeasible solutions = 0

Total number of solutions = 1

$PFS$ = 1

Note

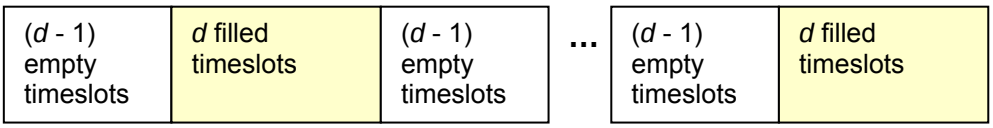
- Case 1 and Case 2 have *SR* = 0, which means that all the timeslots are fully utilized. However, the size of the feasible solution space has been drastically reduced from 24 (Case 1) to 2 (Case 2).
- Case 3 has *SR* = 1 / 4 = 0.25, which means that there is one extra timeslot, which is a wastage. This could account for a better probability of success as compared to cases where *SR* = 0.
- Case 4 is trivial and will not happen in any timetabling problem. Since there is only one solution, there is no need to work on optimization.

Table 6. *PFS* and *SR* against varying class durations and available timeslots

| | *N* = 4 | | *N* = 5 | | *N* = 6 | | *N* = 7 | | *N* = 8 | | *N* = 9 | | *N* = 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | *PFS* | *SR* | *PFS* | *SR* | *PFS* | *SR* | *PFS* | *SR* | *PFS* | *SR* | *PFS* | *SR* | *PFS* | *SR* |
| *d* = 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| *d* = 2 | 0.5 | 0 | 1 | .2 | .1 | 0 | 1 | .14 | .14 | 0 | 1 | .11 | .22 | 0 |
| *d* = 3 | 1 | .25 | 1 | .4 | .33 | 0 | 1 | .14 | 1 | .25 | .13 | 0 | .57 | .1 |
| *d* = 4 | 1 | 0 | 1 | .2 | 1 | .33 | 1 | .43 | .25 | 0 | .6 | .11 | .88 | .2 |
| *d* = 5 | - | -ve | 1 | 0 | 1 | .17 | 1 | .29 | 1 | .38 | 1 | .44 | .2 | 0 |
| *d* = 6 | - | -ve | - | -ve | 1 | 0 | 1 | .14 | 1 | .25 | 1 | .33 | 1 | .4 |
| *d* = 7 | - | -ve | - | -ve | - | -ve | 1 | 0 | 1 | .13 | 1 | .22 | 1 | .3 |
| *d* = 8 | - | -ve | - | -ve | - | -ve | - | -ve | 1 | 0 | 1 | .11 | 1 | .2 |
| *d* = 9 | - | -ve | - | -ve | - | -ve | - | -ve | - | -ve | 1 | 0 | 1 | .1 |
| *d* = 10 | - | -ve | - | -ve | - | -ve | - | -ve | - | -ve | - | -ve | 1 | 0 |

Based on the same working principles noted from Case 1 to Case 4, Table 6 shows an extrapolation of *N*, the total number of available 1-hour timeslots, from 4 to 10. To elucidate the effect of classes having durations that are 1 to 10 timeslots long, several conclusions could be drawn from the table.

- The first obvious conclusion is that fewer classes are allowed as the class duration is increased.
- If the total number of available timeslots (*N*) is a multiple of the class duration, there will be no slack in terms of available timeslots and *SR* = 0. In this case, the *PFS* decreases due to fragmentation.
- If the total number of available timeslots (*N*) is not a multiple of the class duration (*d*), there will be slack and *SR* > 0. This represents a waste of the precious resources, or unused available timeslots. However, because of the slack, the *PFS* is 1 or close to 1. This implies that it is easier to find a feasible solution.
- An interesting observation is the amount of extra timeslots needed such that *PFS* = 1 (no infeasible solution). As seen in Figure 10, this number of extra timeslots can be computed as (*m* - 1) (*d* - 1), with *m* classes of duration *d* timeslots each. This arrangement is the worst in terms of having the highest number of the largest fragments.




Size of each fragment = (*d* - 1) timeslots
Total number of fragments = (*m* - 1)
Total number of empty timeslots = extra timeslots = (*m* - 1) (*d* - 1) timeslots
Total number of timeslots needed = (*m d*) + (*m* - 1) (*d* - 1) timeslots

Figure 10. The worst case arrangement of classes with the highest number of the largest fragments.

Final notes

- Increase in class duration will reduce the number of classes.
- Increase in class duration will make it more difficult to find a feasible solution unless a minimum amount of extra timeslots are added as slack.

## 3.7. Reducing the Effect of Class Interactions

In EEE, there is a combination of classes with different durations. What effect does this have on the feasible solution space? To study the effect, we could first classify classes according to their duration. We could then identify the following cases according to their manner of interaction and try to elucidate their effect. Two types of classes are termed as having an interaction if there is a hard constraint stating that they are not allowed to occupy the same timeslots.

Case 1 (Absence of hard constraint interaction)

There is no interaction among the different types of classes. In this case, each type of classes can be scheduled independently. The timetabling problem is being reduced to a series of smaller sub-problems. A class scheduled on a timeslot will not stop another class of a different duration to be scheduled.

The total number of timeslots required is the maximum of all the timeslots required by all the sub-problems. However, if the number of

timeslots has been prefixed, then depending on the number of classes and their duration, the *PFS* for some of them may be low.

Case 2 (Presence of hard constraints interaction)

There is interaction between the types of classes. In other words, allocating different types of classes on the same timeslot is not allowed. Hence, the total number of timeslots required is the sum total of all the timeslots required by each type of classes. In addition, the order the classes are allocated becomes important. This is explained as follows.

Let us assume that there are two classes $C_1$ and $C_2$. The duration for $C_1$ and $C_2$ is 1 and 2 timeslots respectively. They are not allowed to overlap. Hence, the required number of timeslots is 3. There are two possible ways to allocate the classes. One way is to allocate $C_1$ first and followed by $C_2$ while the other is to do the reverse.

Figure 11 shows the first way. Note that there is an infeasible solution and the *PFS* = 2 / 3 = 0.67.

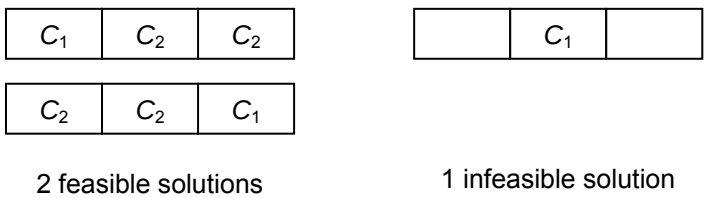
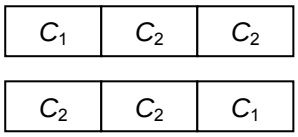



Figure 11. Allocate $C_1$ first and then $C_2$, *PFS* = 2 / 3 = 0.67

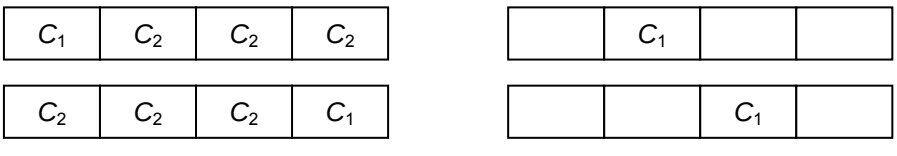
Figure 12 shows the second way. Note that there is no infeasible solution. Hence *PFS* =1.




2 feasible solutions

Figure 12. Allocate $C_2$ first then $C_1$, *PFS* = 1

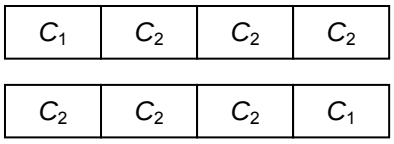
Next the duration of $C_2$ is increased to 3 timeslots. Figure 13 and Figure 14 show the difference in their solution spaces and hence the *PFS*. Again, the same observation is noted. If the class with a shorter duration is scheduled first, there is a possibility of producing an infeasible solution, but if the longer class is scheduled first, there is no infeasible solution. Instead *PFS* has a value of 1.




2 feasible solutions

2 infeasible solutions

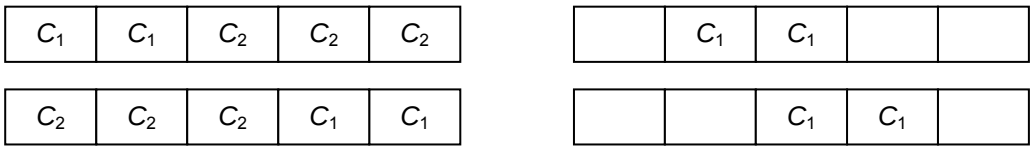
Figure 13. Allocate $C_1$ first and then $C_2$, *PFS* = 2 / 4 = 0.5




2 feasible solutions

Figure 14. Allocate $C_2$ first then $C_1$, *PFS* = 1

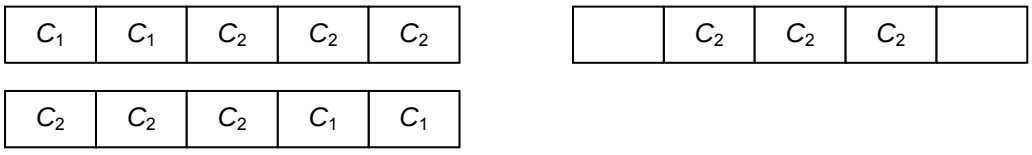
Finally, the duration of $C_1$ is increased to 2 timeslots. The solution spaces are shown in Figure 15 and Figure 16. There is a slight difference in this case as compared to the earlier cases. Figure 16 shows that infeasible solutions could not be pre-empted anymore by scheduling longer class first. However, the *PFS* is still better than doing the reverse.




2 feasible solutions

2 infeasible solutions

Figure 15. Allocate $C_1$ first and then $C_2$, *PFS* = 2 / 4 = 0.5




2 feasible solutions

1 infeasible solution

Figure 16. Allocate $C_2$ first and then $C_1$, *PFS* = 2 / 3 = 0.67

Overall conclusions

The following conclusions could be drawn based on the observations above.

- The required number of timeslot for classes with different durations and without any hard constraint on one another is that of the longest class.
- The required number of timeslots for classes with different durations and not allowed to overlap (hard constraint effect) is the sum total of the timeslots required by each class.
- To increase the probability of obtaining a feasible solution, it is better to schedule classes with the longest duration first, followed progressively by the shorter classes.

# Chapter 4 A Novel EA to Timetabling

Formulating the timetabling problem as a permutation problem fits well into the paradigm of an evolutionary algorithm.

Every evolutionary algorithm starts with the generation of an initial population. This is analogous to allocating a set of classes into the set of timeslots, albeit not just one allocation but a set (commonly called an initial population) of allocations. Each of these allocations is commonly termed as a chromosome.

Once the initial population is generated, the second stage of the algorithm is to select an individual chromosome from the population to undergo a series of evolutionary transformations (commonly known as operations). The main evolutionary operations are the crossover and mutation. Whichever way the operators are designed, the outcome is always another permutation of the same set of classes and timeslots. Repetition and omission are strictly not allowed in a permutation. The use of evolutionary operations may produce another permutation, at times a better one.

An evolutionary algorithm is a powerful stochastic algorithm for searching a globally optimal solution in a search space [25]. A timetabling problem has a vast and complicated solution space

containing both feasible and infeasible solutions. As a search problem, it is most apt to be solved by an evolutionary algorithm.

## 4.1. Co-evolution

Having established that an evolutionary algorithm is a suitable approach to solve the timetabling problem, the next issue is to design the evolutionary algorithm.

Since there are three types of classes, differentiated by their duration: 1 hour, 2 hours and 3 hours, a possible approach is to design a co-evolutionary algorithm [43, 47]. Each type of classes is treated as an individual species and co-evolves at the same time. The species survival depends on the way they influence one another, either competitively or cooperatively. It has been found that this is not the best approach. The algorithm takes a long time to evolve because of the strong interactions between the types of classes. This is especially so between the lecture classes (1 hour long) and the practical classes (3 hours long), which are not allowed to overlap. This will result in the generation of many offspring populations of co-species that will clash with one another over the use of timeslots. The system then expends a considerable amount of time trying to resolve these problems.

## 4.2. Multi-stage Co-evolutionary Algorithm (MSCOA)

The best way to deal with the generation of infeasible solutions is to pre-empt it. As noted earlier, generation of infeasible solutions could be pre-empted by ordering the timetabling process according to the different duration of classes.

Each type of classes will still evolve on its own but it must wait for its turn. The order is such that classes with the longest duration will be evolved first and followed progressively by the shorter classes. This approach has produced good results as seen in [46].

## 4.3. Mutation and Crossover Operation

Most evolutionary algorithms use both the crossover and mutation operation. For a permutation problem such as the examination timetabling problem, using only the mutation operation had been found to be sufficient as seen in [2, 39, 42]. The following discussion shows that a crossover operation is very similar to a mutation operation. The partially matched crossover technique (PMX) is used, as suggested by Goldberg and Lingle [27] for a permutation problem.

Figure 17 depicts two chromosomes selected from a population of chromosomes. Each chromosome is represented as a string of genes. Each gene could assume a number, chosen from 1 to 10. These chromosomes behave as parents and they are about to mate by

performing a crossover operation. One or two crossover points have to be defined before the crossover operation can proceed. Figure 17 shows that two crossover points are defined randomly over the strings. The crossover operation is performed in two stages.

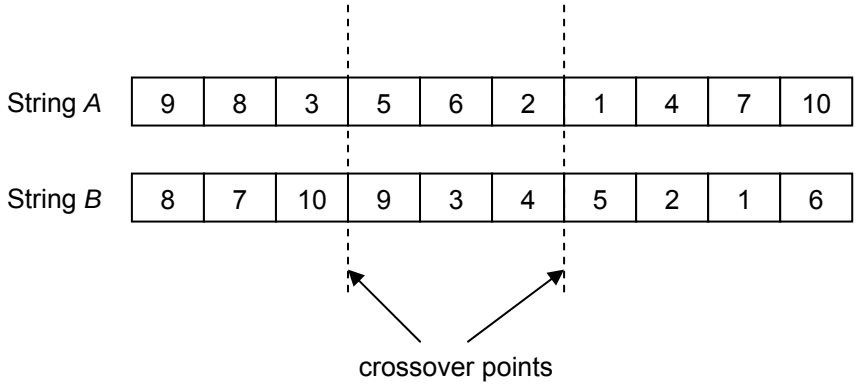



Figure 17. Parent strings *A* and *B* before crossover

Figure 18 shows the first stage of the crossover operation. The genetic material within the crossover points of one parent string (string *A* of Figure 17) are exchanged with the genetic material within the same crossover points of the other parent string (string *B* of Figure 17). The resultant offspring strings contain many duplicates. For example, the number 9, 3 and 4 in string $A_1$ are repeated and the number 2, 5 and 6 are missing. In a permutation problem such as a timetabling problem, this is not acceptable.

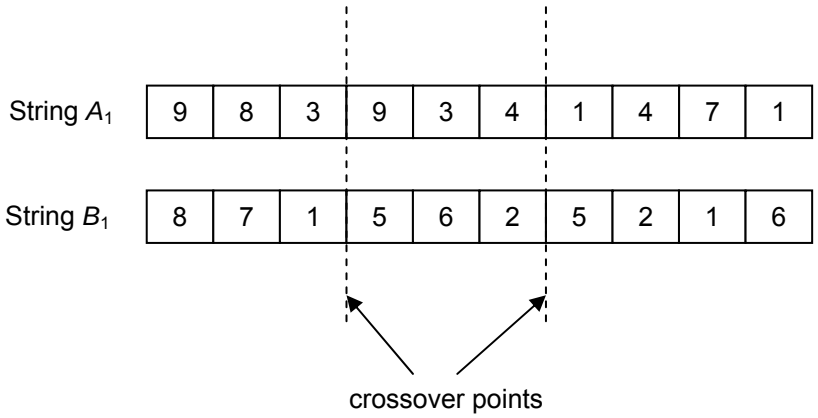



Figure 18. String $A_1$ and $B_1$ after an exchange between two crossover points

Stage two of the crossover operation is to remove the duplicated genes and restore the missing genes. PMX dictates that what has been changed in between the crossover points stay. The duplicated gene outside the crossover window will be replaced by what has been displaced by this gene within the crossover window. In this way, all the duplicated genes will be removed and all the missing genes will be restored. This is clearly explained in Figure 19.

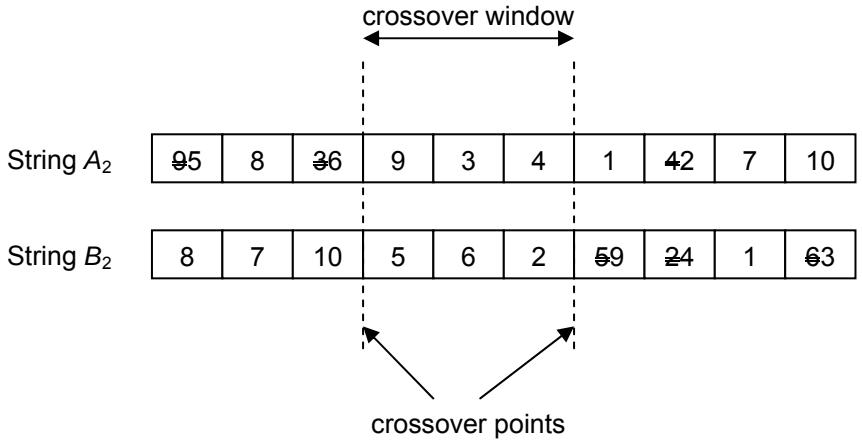



Figure 19. String $A_2$ and $B_2$ after replacing duplicated genes with missing genes

The crossover operation has effectively transferred a substring from one parent string to another parent string. However, in doing so it has disturbed the string pattern outside the crossover window. The wider the crossover window, the wider is the effect. The resultant string may actually resemble the other parent string that has undergoing some controlled form of mutation. This can be seen in Figure 20.

Comparing string *A* and string $A_2$ (after crossover), there are 6 positions where changes have occurred. String $A_2$ (offspring) does not resemble string *A* (a parent) very well. On closer observation, it resembles an offspring of the other parent (string *B*) that has undergone mutation within a window, which is outside the crossover window. We can therefore define this outside window as a mutation window.

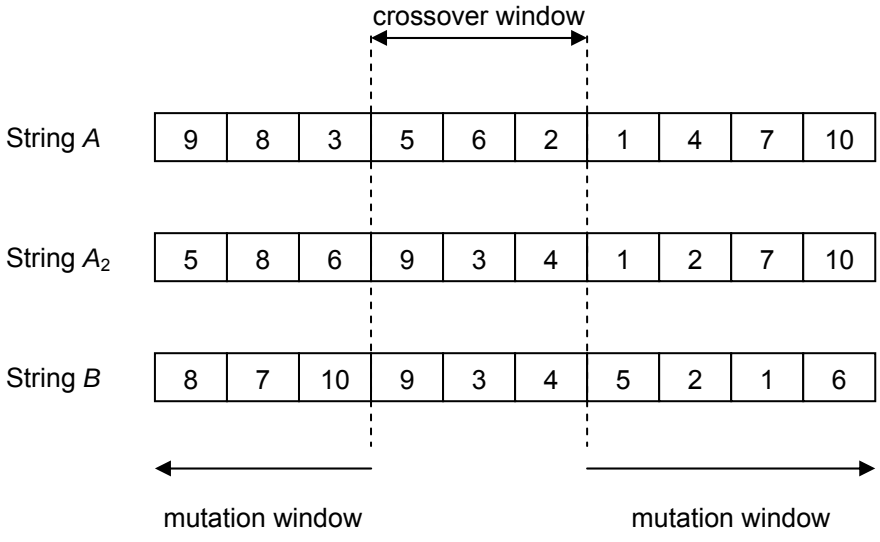



Figure 20. Similarity between a crossover and a mutation operation

Instead of defining a crossover window, we can define a mutation window, and instead of performing a crossover operation, a mutation operation is performed within the mutation window. The effect of the two operations is identical. However, a mutation operator is much easier to design and its execution is simpler and more efficient. Furthermore, in defining a mutation window, there is a better control over the amount of a chromosome that needs to be changed.

## 4.4. Directed Window Mutation (DWM)

A novel mutation method is used for the proposed evolutionary algorithm for solving EEE curriculum timetabling problems. This method is a result of observing the similarities between a crossover and a mutation operation.

This method begins by defining a mutation window of a certain size over a small portion of a chromosome. The mutation window is randomly placed over the chromosome. The earlier allocation of classes to timeslots within this window is first released. The classes are then reallocated randomly back into the timeslots within the same window. In this way, the allocation of classes that are outside this window is not disturbed. This is desirable for certain permutation problems such as a timetabling problem that favors certain sub-string pattern. A typical

example of a favorable sub-string pattern is a string of 3 consecutive lecture classes for the same lecture group.

A large mutation window will create a large perturbation to a chromosome. This is crucial in the early stages of an evolution, as it allows the algorithm to make big steps over the search space, thus providing a diverse collection of chromosomes in a population. Secondly, it serves as a mean to escape from being trapped in a local optimal point. However, a small mutation window is preferred towards the end of an evolution. This is like making small steps around an area, which is similar to a local search strategy such as the hill-climbing operation. A good strategy is to progressively reduce the size of the mutation window with the iterative process.

To ensure that the evolution is well directed toward a global optimum, we need to provide some heuristic guide to the direction of mutation. After performing mutation on a population, the best chromosome of the offspring population (new best) is compared against the best chromosome of the parent population (previous best). If the previous best chromosome is better than the new best chromosome, the previous best will replace the worst chromosome of the new population. In this way, the best chromosome will always survive through all the generations. Its presence serves to direct the evolution through mutation to search for the presence of a better solution.

In a real application, the total length of a chromosome is computed by adding all available resources in each timeslot. A window is computed based on a certain percentage over it. The starting point of the window is then randomly chosen over the length of the chromosome. Wrap around to the front may happen if the end point of the window is greater than the end of a chromosome.

## 4.5. A Multi-Stage Co-evolutionary Algorithm with DWM

In summary, the proposed evolutionary algorithm for a curriculum timetabling problem will incorporate all the concepts that have been developed through the above discussions. These concepts are enumerated below.

- Classes are grouped into types based on their duration. Each type of classes will constitute a sub-schedule, like that of a sub-species in a co-evolutionary environment.
- To build an initial population, the sub-schedule of classes that have the longest duration is scheduled first. This is then followed by the second longest duration sub-schedule and so on until all the sub-schedules are scheduled.
- The same order is imposed when it comes to mutation.
- Only directed window mutation is used at each evolutionary step.

Although the algorithm is developed for curriculum timetabling, it can also be used to solve other types of university timetabling.

The design, implementation and application of the algorithm will be discussed in subsequent chapters.

# Chapter 5 Design and Implementation Issues

The key building blocks of an evolutionary approach consists of the development of an evolutionary algorithm, structuring an appropriate coding scheme, building of an initial population, design of evolutionary operators, formulation of a fitness function, adoption of a selection scheme and defining a termination criterion. The design and implementation of these building blocks depend a lot on the application. This chapter highlights the design and implementation issues of the algorithm and the coding scheme. The other building blocks will be discussed in the following chapters. The curriculum timetable, being the most complex, is used to illustrate some of these issues.

## 5.1. Algorithm

Figure 21 shows the main flow chart of a generic evolutionary algorithm. It depicts the minimum set of building blocks. However, the design and implementation of each building block varies according to the application. Figure 22 shows the proposed main flow chart of the evolutionary algorithm used in solving the curriculum timetable.

The proposed algorithm takes into account the hard constraint interactions of the sub-schedules. The intuitive way to resolve this undesirable effect of the interactions is to schedule classes with longer

duration first. This is crucial in generating an initial population of feasible solutions. As mentioned earlier, such feasible solutions may not be optimal. Subsequent evolutionary generations will then seek to improve the fitness value of this initial population. In each of the evolutionary generations, classes with longer duration are also evolved first.

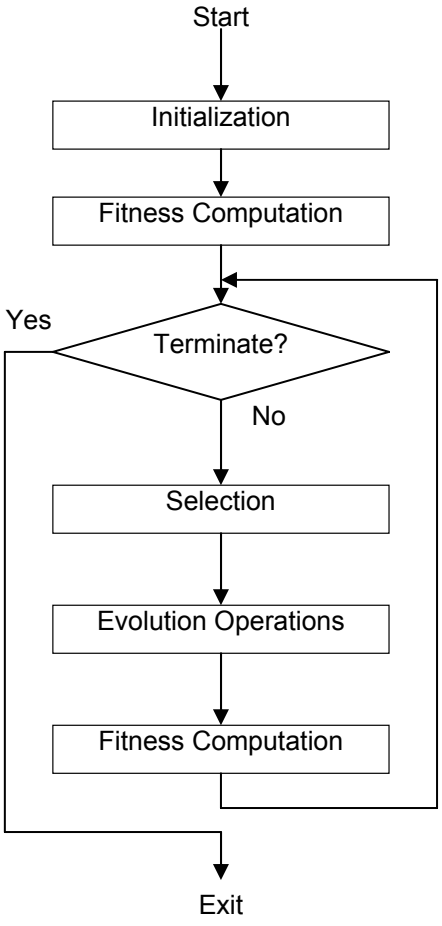



Figure 21. Generic EA

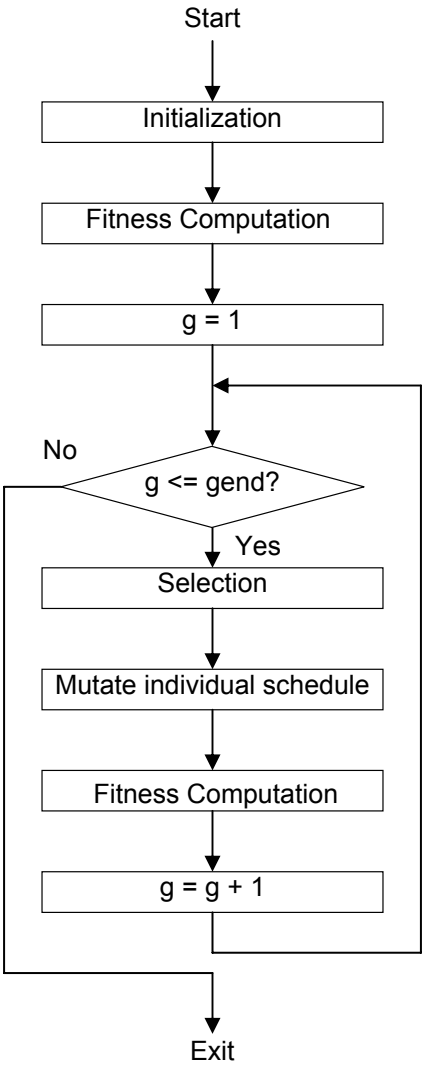



Figure 22. Proposed EA for Curriculum Timetabling

## 5.2. Coding

Choosing a proper coding for an application is very important to the running of an evolutionary algorithm [55]. The two most common

methods are the direct and indirect method. An indirect method is to code the problem domain using a binary system. The conversion is usually very complicated, but the design of the evolutionary operators is rather straight forward. It involves flipping “0” and “1”. Another way is to code the problem domain directly and although this is simpler, the evolutionary operators become rather involved. As a timetabling problem involves classes and timeslots which are difficult to code into binary strings, the indirect method is often used.

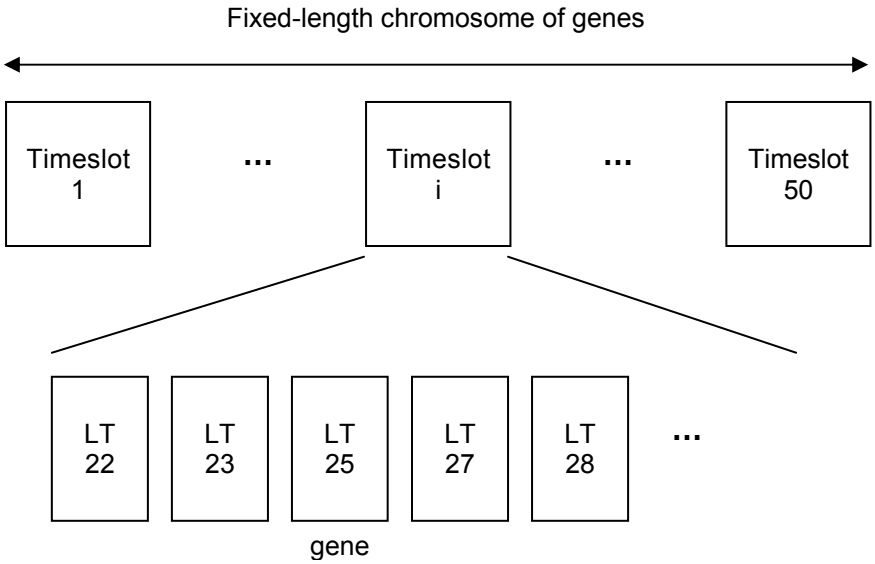



Figure 23. Coding of a Lecture Schedule

Figure 23 shows a direct coding of a lecture schedule. There are 50 timeslots in a week. For every timeslot, there are a limited number of lecture theatres (LTs) available for lecture classes. This information is predetermined by the University’s Registrar Office. As seen in the figure, chromosome for the lecture schedule is defined as a string of available LTs for each timeslot in a week. Since the number of LTs available in

each timeslot is fixed, the total length of a chromosome is fixed. The alleles for these genes are the set of lecture classes.

This is repeated for the other sub-schedules with their types of teaching facilities. A chromosome for a complete timetable is the sum total of all these chromosomes. A population consists of a set of these timetable chromosomes.

# Chapter 6 EEE Examination Timetable

An examination timetable allocates the corresponding examinations of the various subjects over the total examination period. This chapter describes an automated examination timetabling system based on a stochastic search methodology, namely an evolutionary algorithm augmented with a domain specific local search strategy. The proposed system is not developed by the author but was developed by a group of final year students, co-supervised by the author [52]. The conception of that project was done prior to the commencement of his present work. It is mentioned here to present a complete view and some of the shortfalls are taken into consideration in the present work.

The current approach in EEE is to manipulate manually the previous year's examination timetables, taking into account impending changes. However, with the introduction of the modular system and the increasing number of students, such an approach is no longer efficient. Some students are also taking a combination of subjects that would not have been feasible in the past. The increase in student intake and the number of subjects offered as well as the possibility of students repeating lower year subjects have greatly complicated the timetabling process. In addition, the examination timetable planned by the school must be iterated and fine-tuned at the university level among all the schools to satisfy the overall seating capacity of all the examination halls. Hence,

there arises a need for an automated system. The resultant system should deliver a better quality examination timetable as well as a shorter planning time.

The automation of examination timetable planning is very well explored, especially that of an evolutionary approach and will not be covered in detail in this thesis. The examination timetabling system will only be briefly mentioned in this chapter. It is based on an evolutionary approach augmented with a domain specific local search strategy. This approach has been found to be effective in producing very good results [52]. Specialized evolutionary operators have to be developed to suit the examination timetable in EEE. They are designed with expert knowledge from the timetable planner.

## 6.1. Hard Constraints

As in all timetabling problems, there are both the set of hard and soft constraints, unique to an institution, to be satisfied. There are two hard constraints that are universal to all examination timetabling problems. They are:

- No student can be in two or more examination halls at any one period;
- There must be sufficient seating capacity in the examination hall for all the registered students scheduled in any one period.

In view of these constraints, a number of subjects can be scheduled in one examination period as long as the seating capacity of the available examination halls of that period is not exceeded. Violation of this hard constraint will result in an infeasible timetable. When a large group of students reads a particular subject, which is very common in EEE, they will have to be divided into a few examination halls and be examined together.

To simplify the problem, the algorithm consults the curriculum timetable during scheduling. Based on the curriculum timetable, subjects with lecture classes being conducted in the same period in the curriculum timetable will be scheduled together in the same examination period as well. This helps to shorten the total examination period.

The evolutionary operators such as the crossover, mutation and local search operators are carefully designed to avoid any violation of these hard constraints.

## 6.2. Soft Constraints

The key soft constraint is the rest time between two examinations which are scheduled in two separate periods. The rest time may be 0.5 day, 1 day, 1.5 days or more days apart from each other. The reason for having this constraint is to allow sufficient time for the students to prepare their examination. It is preferable to have two or more days

apart between any two examinations taken by the same student. However, the total number of days allocated for the various examinations cannot exceed the total examination period set by the University.

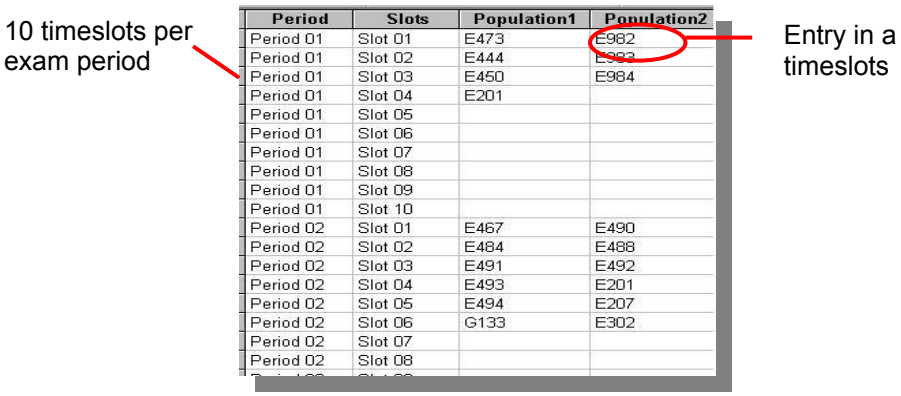
Other types of soft constraints include:

- Avoid examination periods outside normal office hours.
- Avoid examination papers of different scheduled end-time in the same examination hall. Group of students leaving a hall ahead of another group of students sitting for a longer paper is a distraction to those remaining.

The amount of soft constraint violations is used to measure the quality of the examination timetable. This is very hard to ascertain in the manual system. Due to a short planning period, the planner accepts the first successfully completed timetable. The examination timetable is planned before the students register for their choice of subjects. This is different from other universities. A poorly designed examination timetable will affect the registration of certain subjects since examination timetables are released during subject registration. Some subjects will have fewer students than projected, as students prefer subjects that are not close to each other. This would in turn affect other subjects in the current and subsequent semester. Therefore, in automating the examination timetabling process, it is important that a mean to measure the quality of an examination schedule be established.

## 6.3. Algorithm

The proposed algorithm incorporates domain knowledge related to the problem and is used to fine-tune the process of evolution. This kind of algorithm has been described as a memetic algorithm [2, 36, 53], which describes a meme as an idea or concept that is passed around a society. Individuals can then adapt this idea to suit their own environment. To apply this algorithm to a problem, a direct coding scheme with an appropriately defined set of evolutionary operators is used.




| Period | Slots | Population1 | Population2 |
|---|---|---|---|
| Period 01 | Slot 01 | E473 | E982 |
| Period 01 | Slot 02 | E444 | E983 |
| Period 01 | Slot 03 | E450 | E984 |
| Period 01 | Slot 04 | E201 | |
| Period 01 | Slot 05 | | |
| Period 01 | Slot 06 | | |
| Period 01 | Slot 07 | | |
| Period 01 | Slot 08 | | |
| Period 01 | Slot 09 | | |
| Period 01 | Slot 10 | | |
| Period 02 | Slot 01 | E467 | E490 |
| Period 02 | Slot 02 | E484 | E488 |
| Period 02 | Slot 03 | E491 | E492 |
| Period 02 | Slot 04 | E493 | E201 |
| Period 02 | Slot 05 | E494 | E207 |
| Period 02 | Slot 06 | G133 | E302 |
| Period 02 | Slot 07 | | |
| Period 02 | Slot 08 | | |

Figure 24. Two populations of timetable solutions

Figure 24 shows a portion of two populations of solutions. Each population consists of a set of feasible solutions. A new offspring population is generated at the end of an evolutionary step. An examination period is defined as the time allocated for students to write their answer scripts. At EEE, each examination period is of 2 to 3 hours

long. There are two periods available in each calendar day of a working week, one in the morning and the other one in the afternoon. A maximum of 10 subjects is allowed in each examination period, hence there are 10 timeslots in each period. The total number of timeslots forms a chromosome. These timeslots are arranged chronologically. In each timeslot, there are a number of examination halls available. An examination hall is either left vacant or be scheduled for an examination. The solution is directly represented as a string array, where each subject code represents a subject scheduled in a timeslot at a particular examination hall.

Figure 25 shows the flowchart of this algorithm. Evolution begins with a population of chromosomes. During reproduction, crossover of genes from a pair of existing chromosomes (parents) to form a new pair of chromosomes takes place. These newly created chromosomes (offspring) can then be mutated. At the end of each evolutionary step, the fitness of an organism is computed. A population with a high fitness value has a better chance of survival in the next generation through the selection process.

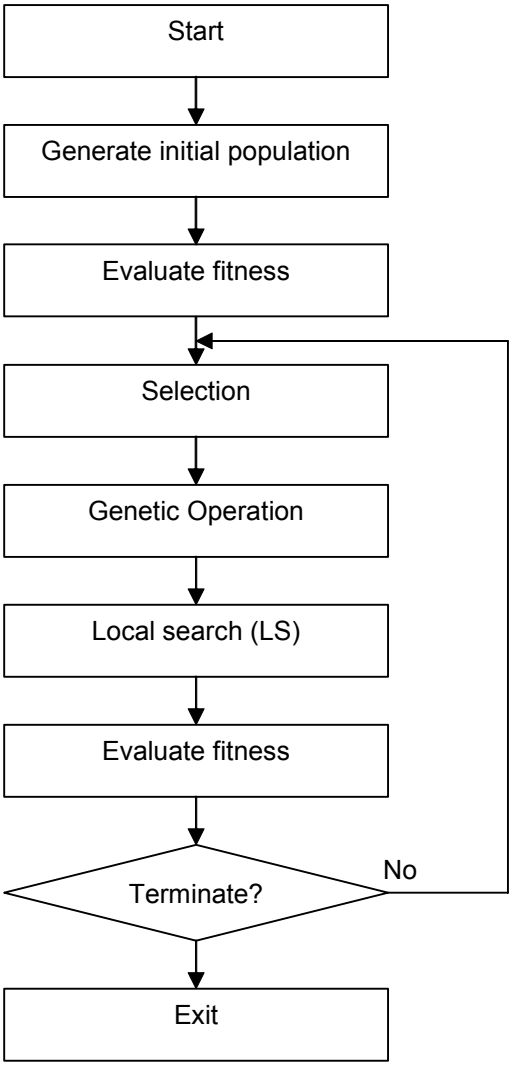



Figure 25. EA for Examination Timetabling

## 6.4. Fitness Function of an Examination Timetable

A simple weighted fitness function is used to assess the fitness of each solution in a population. It is computed as a weighted composite of four different groups. They are defined as follows.

- Subjects that are in the same semester. Most students would take this group of subjects ($G_1$).
- Subjects that are one semester apart. These apply to a student who is reading a subject 1 semester ahead or re-reading a subject belonging to the previous semester ($G_2$).
- Subjects that are two semesters apart. This implies that a student has read subjects that are for different year of study ($G_3$).
- Subjects that are three semesters apart ($G_4$).

Each of these composite groups contributes to the overall fitness differently. $G_1$ has the highest weight ($w_1$ = 100) since it involves subjects taken by students in the same year and semester. A user can also define the weights ($w_2$, $w_3$, $w_4$) for the other groups interactively. Experimentally, $w_1$ = 100, $w_2$ = 30, $w_3$ = 20 and $w_4$ = 10 give good result.

Penalty functions are introduced to measure the rest time between two successive examinations that are conducted for subjects in the same group ($G_1$) and for subjects that are conducted one or more semesters apart ($G_2$ to $G_4$).

The fitness function of each timetable solution is given as

fitness = (max_fitness – total_penalty) / max_fitness,

where total_penalty is the actual penalty incurred by all the groups ($G_1$ to $G_4$) and max_fitness is the maximum fitness without any penalty.

The aim of the algorithm at each step of evolution is to select solutions with better fitness as compared to others in the current population of solutions. Evolution terminates when the fitness of solutions converge or a maximum number of steps has been tried.

## 6.5. Domain-specific Local Search Strategy

Conventional evolutionary operators for generation of initial population, crossover, mutation and selection are used in the proposed algorithm. A simple local search operator [42], also known as hill climbing, is applied after each mutation to shift the solution to the nearest optimum. During the process, each subject is chosen randomly to be relocated. The period with the lowest penalty is chosen for relocation. This is repeated until no improvement can be made as outlined below:

- Repeat
    - Randomly select a subject.
    - Calculate the penalty for all the examination periods.
    - Schedule the subject into the best location.
- If there is an improvement then repeat the whole process until all the subjects are selected

## 6.6. Results and Discussions

Table 7. Comparison of penalties of the manually planned and computer generated examination timetable

| Manually planned | | | Computed generated | | |
|---|---|---|---|---|---|
| Subject (from - to) | Day difference | Penalty | Subject (from - to) | Day difference | Penalty |
| E201 – E202 | > 3 | 0 | E201 – E204 | > 3 | 0 |
| E202 – E204 | 1.5 | 35 | E204 – E205 | 1 | 65 |
| E204 – E227 | 1 | 65 | E205 – E120 | 2.5 | 2.5 |
| E227 – E206 | 2.5 | 2.5 | E120 – E206 | 1.5 | 35 |
| E206 – E203 | 0.5 | 82.5 | E206 – E227 | 1.5 | 35 |
| E203 – E208 | 0 | 100 | E227 – E207 | 1 | 65 |
| E208 – E205 | 1 | 65 | E207 – E202 | 1 | 65 |
| E205 – E207 | 1 | 65 | E202 – E208 | 1 | 65 |
| E207 – E120 | 1 | 65 | E208 – E203 | 1 | 65 |
| Total | | 480 | Total | | 397.5 |

Two examination timetables are generated for Semester 1, 2000/2001 academic year. One was manually planned while the other was generated by the proposed algorithm. For simplicity, only second year subjects belonging to the $G_1$ group are highlighted to demonstrate the effectiveness of this algorithm as well as the computation of the penalty function. It takes about 4 hours to complete each simulation.

Table 7 compares the penalties incurred by this highlighted group of subjects in the two examination timetables, which shows a 17% reduction in penalty for the computer generated examination timetable.

The overall fitness of the manually planned examination timetable is 73% while that of the computer generated is 93%. Therefore, the

proposed algorithm does generate a better examination timetable as well as provide automation for the examination timetabling process.

## 6.7. Conclusions

An evolutionary algorithm with local search has been successfully implemented to plan the examination timetable of EEE. The resultant timetable has a fitness value of 93%. This is much better than that of the current manual method, which has a fitness value of 73%. The application of a local search operation in addition to a genetic evolution has indeed improved the final solution, regardless of the initial population of chromosomes. This is highly desirable.

The process of scheduling an examination timetable has been automated, which, up to now, has been carried out by a human scheduler. By adjusting the definition of fitness/penalty of a timetable, different versions of examination timetables can be easily generated. This will encourage more experimentation and provide more flexibility in timetable parameters such as resources, course structure and student intake. This translates to better examination preparation time for students and an easier task for examination timetabling.

Though this is a much simpler case of timetabling as compared to that of a curriculum timetable, it has provided much insight into a more general approach to timetabling.

One of the key points to note is that it is very difficult to design a crossover operator. The problem is with the frequent occurrence of many hard clashes, which is very costly to fix. Coupled with this is the high cost of incorporating a domain-specific local search operation. These considerations give rise to the idea of using a modified mutation operator.

The second major problem is the duration difference in some of the examination papers. It is not good to mix such examination papers in a single examination hall and this complicates the process of generating the initial population and the design of an efficient evolutionary operator. It gives rise to the idea of a multi-stage co-evolution, which is proposed in this thesis.

It has also been found that a direct coding scheme is suitable for the examination timetabling problem. An indirect scheme, usually in a binary format, had been tried and was found to be very difficult to work with. A lot of meaningless chromosomes was generated at every step, and was difficult to handle. This problem is also found in other types of timetabling problem. As such, direct coding is adopted for timetabling problems in EEE.

# Chapter 7 EEE Curriculum Timetable

The main task of a university curriculum timetable is to schedule teaching activities such as lecture classes over a set of timeslots. The complexity of a curriculum timetable depends on the curriculum structure of a university and its unique requirements. These are expressed in the number and types of classes as well as the effect of hard constraint interactions among the classes. These complexity factors are exemplified in the EEE curriculum timetable. Since EEE curriculum timetable has a very high level of complexity, solving it will provide valuable insights. This will help to formulate a general solution method to other curriculum timetabling problems with various levels of complexities.

A multi-stage co-evolutionary approach as developed in earlier chapters is used to solve the curriculum timetabling problem. Although the idea of decomposing a timetabling problem had been proposed [39, 48], there are some major differences. The problem addressed was that of an examination timetable and there is only one type of examination paper, namely the written examination. However, there is more than one type of teaching components or classes in a curriculum timetable. As a result, more constraints need to be satisfied. The problem of avoiding any violation of constraints in constructing a feasible solution is more acute here. The approach used in resolving the effect of these hard

constraints is to decompose a timetable into a number of sub-schedules. This means that one schedule is planned for each type of teaching classes. These sub-schedules are to be co-evolved in a multi-stage order.

## 7.1. Hard Constraints

The hard constraints for curriculum timetabling are similar to that of examination timetabling. They are:

- No student can be in two places at any one period;
- There must be sufficient seating capacity in the venue for all the registered students scheduled in any one period.

There are additional hard constraints that are unique to EEE. Lecture classes conducted for a subject in one lecture group should not clash with lecture classes in other lecture groups for the same subject. In addition, a particular lecture class conducted for a subject should not be scheduled on the same day for another lecture class for the same subject within a lecture group unless the subject requires consecutive lectures. Practical subjects such as the laboratory classes must have at least an hour break. This allows time for the technicians in the laboratory to break for lunch and to prepare for the next class.

## 7.2. Soft Constraints

Soft constraints are also preferences, which may be violated. However, a timetable with too many such violations is considered as poor in quality. The number and type of soft constraint violations are used to measure the fitness value of a timetable.

The set of soft constraints or preferences are:

- Classes allocated to timeslots that are outside the normal office hours. These timeslots are not well liked by staff and students. The number of such allocations should be minimized.
- Consecutive lecture classes are allocated where possible for the same lecture group. If lecture classes are allocated with too many free timeslots in between, the time spent by students waiting for the next lecture class may not be well utilized.
- Consecutive lectures are allocated the same lecture theatre to minimize the need for students to move between lecture theatres.
- Finally, students generally prefer lunch period to fall between 1130 and 1330.

## 7.3. Evolutionary Approaches

Many evolutionary approaches applied the conventional genetic algorithm augmented with several enhancements. One such approach is to use only the mutation operation coupled with local search [52].

This was the approach used in the early attempts to solve the EEE examination timetabling problem. However, there are several problems unique to the curriculum timetable and they are not easy to solve. There are more than one components or types of classes, each with different duration. There are also multiple lecture groups for the same lecture class. Due to these differences, it is difficult to generate an initial population of feasible solutions (i.e. free of any violation of hard constraints). The other problem is in adapting the evolutionary operators such that only offspring that represent feasible solutions are produced. Apparently, this approach is not very effective and may not be able to provide a feasible solution [32, 33].

One other possible approach is to schedule each type of classes separately. Each sub-schedule is treated as a species and a co-evolutionary approach has been considered. Some species with classes of longer duration, once scheduled, will result in having fewer empty timeslots for the other species. Conversely, schedules with classes of shorter duration, once scheduled, will result in having a very fragmented list of empty timeslots for the other schedules, especially compared to those with longer duration (which need a contiguous row of empty timeslots to fit the longer duration). This kind of interaction makes it very difficult for a truly co-evolutionary approach. The same problem, as mentioned earlier, in finding an initial population of feasible solutions is encountered [43]. The co-evolutionary approach though is a good idea has to be modified.

## 7.4. Algorithm

Figure 26 shows the main flow chart of the curriculum timetabling algorithm. It is exactly the same as Figure 22 and is repeated here for ease of explanation. The algorithm takes into account the interactions of the different sub-schedules. The intuitive way to resolve the unpleasant effect of the interactions is to schedule classes with longer duration first. This is crucial in generating an initial population of feasible solutions. These initial solutions though are feasible but may not be optimal. Subsequent evolutions will then seek to improve the fitness value of this initial population. In each generation, classes with longer duration are also evolved first.

## 7.5. Initialization

Each generation of a timetable in the initial population module is actually a sequence of 5 initializations of the composite sub-schedules. The sub-schedule with the longest duration is generated first. This is then followed by sub-schedules for shorter duration classes. When generating a sub-schedule, it is important that there is no violation of any hard constraints within the sub-schedule as well as with those sub-schedules that are generated earlier.

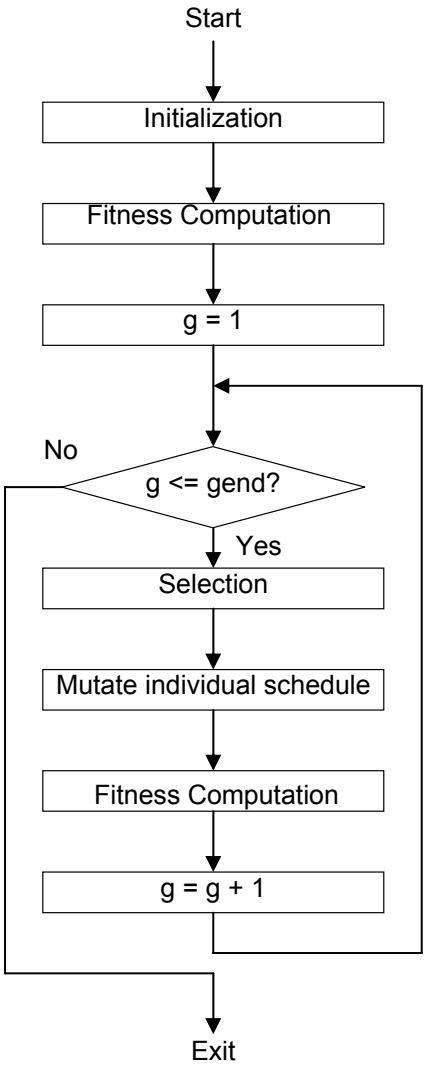



Figure 26. Proposed EA for Curriculum Timetabling

## 7.6. Fitness Computation

The fitness of a timetable *F* is computed based on the satisfaction of the set of preferences or soft constraints, listed earlier in Section 6.2.

*F* is determined by the amount of satisfaction of these preferences.

$$F = a \sum \textit{unoccupied bad slots} +$$

$b \sum$ *consecutive lectures* +

$c \sum$ *consecutive venues* +

$d \sum$ *good lunch hour* +

$e \sum$ *free day*

*Unoccupied bad slots* are computed as the difference between the initial number of available bad slots and the number of the bad slots that are actually allocated to classes by the algorithm.

The weighting factors (*a, b, c, d, e*) are assigned the value of 1 to indicate the fact that all the soft constraints or preferences are deemed as equally important. They can be adjusted according to the need of the timetable planner.

The aim of the algorithm is to maximize *F*, the overall fitness value.

## 7.7. Selection

A classical selection algorithm based on the roulette method [27] is used. Effectively, this method will choose solutions, which are of high fitness values, with greater probability. The set of chosen solutions will then participate in the next process of mutation. Elitism is also added to ensure that the best feasible solution will be retained and participate in

the next evolutionary process. This selection scheme is chosen because of its ease in implementation and the good results generated by other researchers.

## 7.8. Directed Window Mutation

Mutation of a timetable is done sequentially on the individual sub-schedules. The order of execution is duration dependent. A mutation window of certain string length is randomly defined over a sub-schedule. Within it is a sub-sequence of a mixture of filled and empty timeslots. This is like opening up a small window over the whole length of the parent sub-schedule. The length or size of the mutation window is specified as a fraction of the total number of timeslots. Classes that have been allocated in this mutation window are first released. These classes are then randomly reallocated back within the window.

There are several reasons for using a mutation window. A schedule consists of several sub-sequences. The fitness of a solution depends on the arrangement (permutation) of these sub-sequences. It is computed as a summation of the fitness of these sub-sequences. If the mutation is performed randomly across a parent schedule, the probability of producing poorer offspring is high since many good quality sub-sequences would be disturbed. On the other hand, if a small window is defined over the sequence and only the mutation is allowed within this

window, the probability and number of good quality sub-sequences being disturbed are much lower.

Another reason is that this approach is similar to the way a human planner does the planning. The tendency is not to change too drastically an existing working schedule. The planner tends to look for sub-sequences in the existing schedule that can be further improved.

## 7.9. Results and Discussions

This chapter discusses the result obtained on the 2001/2002 EEE curriculum timetable design. A population size of 30 feasible timetables is used. This is found to be a good compromise between speed and diversity. The mutation window of 5% is empirically determined to be the best in terms of convergence speed and diversity. These parameter values are obtained empirically when the algorithm is applied over a smaller number of classes and resources. The number of generations is 1650, as there is no more significant improvement beyond this number. The input and output files for this simulation can be found in Appendix A. It takes about a day to complete the simulation.

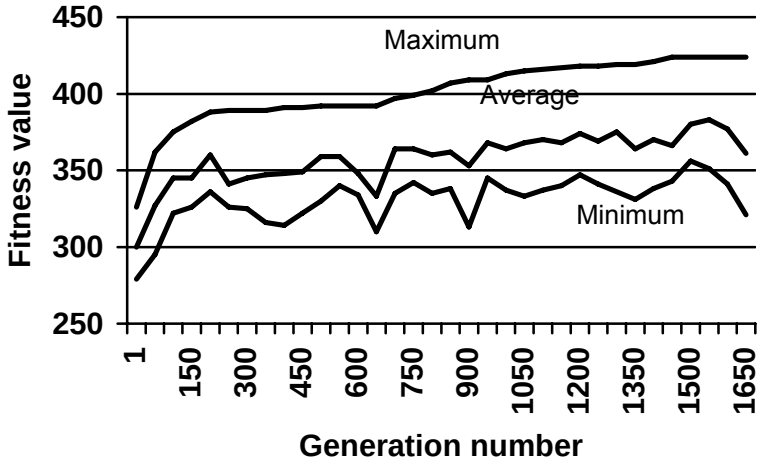



Figure 27. Fitness Value vs Generation

Figure 27 shows the maximum, average and minimum fitness values plotted against the generation number. The trend for the maximum fitness value rises progressively. This is expected as the algorithm restores at least a copy of the previous best timetable at every generation. The trend for the other two fitness values rises too but with some degree of fluctuation. The distance between the maximum and minimum fitness values remain relatively constant in every generation. This provides a good diversity or spread of the fitness quality of a population for the evolutionary algorithm.

As can be seen in Figure 27, there are several local optimum solutions. The algorithm is able to escape from such solutions and continue to seek for better solutions. This shows that the evolutionary algorithm is able to escape a local optimum point, which is the strength of such algorithms.

Table 8 shows the improvement of the maximum fitness values of the initial and the final populations. There is a 30% overall improvement. Almost all the fitness components show some improvement. Since the good lunch fitness component of the initial population has already reached its maximum values, there can be no further improvement.

Table 8. Comparison of Initial and Final Fitness Value

| | Initial | Final | % Improvement |
|---|---|---|---|
| *total* | 326 | 424 | 30% |
| *unoccupied bad slots* | 223 | 265 | 19% |
| *consecutive lectures* | 46 | 66 | 20% |
| *consecutive venues* | 21 | 54 | 16% |
| *good lunch hour* | 36 | 36 | 0% |
| *free day* | 0 | 3 | |

This result is generated based on equal preference on all the individual components of the fitness function (a=b=c=d=e=1, see Section 7.6). As shown in Table 8, the greatest contribution would come from the *unoccupied bad slots* and the least is the *free day* component. Hence, the fitness function is most sensitive to the parameter 'a' and least sensitive to 'e'.

## 7.10. Conclusions

The use of the novel evolutionary approach for complex curriculum timetabling like that of EEE does produce very good results. The algorithm always produces feasible solutions. By using a mutation window of varying sizes, improvement from one generation to the next

can be observed and controlled. Sub-sequences of high fitness values are better preserved. Using elitism in the selection module directs the mutation towards a better quality schedule.

A unique feature of the EEE curriculum timetable, which increases the level of complexity, is the presence of different types of classes. A standard evolutionary approach of using a single schedule is not efficient. A more efficient approach is to treat the type of classes as co-species and co-evolves them in an orderly manner.

The results of testing based on the 2001/2002 curriculum timetable planning have been positive. In the previous manual system, the timetable planner is usually satisfied with the first successfully generated feasible solution even though it may not be optimal. However, with this proposed system, the planner is able to obtain solutions that are both feasible and optimal.

# Chapter 8 EEE Subject Allocation System

In EEE, registration of subjects according to preferences indicated by students is a complex optimization problem. An evolutionary algorithm based system that attempts to maximize satisfaction index (to be defined in Section 8.4) for the overall student group is proposed here. The concept of Directed Window Mutation (DWM) developed earlier in this thesis is used. A further enhancement to the DWM is to vary the size of the mutation window with the generation number. The system involves allocating students' choices of subjects. Since there is only one type of choices, it is not necessary to incorporate the ideas of multi-stage and co-evolution.

Each student is asked to submit two sets of subjects that he/she would like to register for the semester. The problem involves allocating a student's choice of subjects. To facilitate the registration of subjects, the University introduced the Students Automated Registration System (STARS) in 1993. Every subject is given a series of index numbers (e.g., 30 index numbers). Each index number encodes a set of scheduled classes. In STARS, students key in the 4-digit index numbers of the subjects they wish to take. From July 2004, the University has decided to adopt a 5-digit subject index system because the current index has outgrown itself. Designing a registration system that allows students to choose their subjects freely while meeting the course

requirements for majors and minors and satisfying the curriculum and examination schedules becomes a very difficult, complex and challenging task.

STARS accept students' subject choices on a first-come-first-serve basis. This gives rise to a network bandwidth problem when a few hundred students attempt to register their subjects simultaneously. To alleviate this problem, students are divided into several groups and each group is allocated up to two hours to register their subjects. Students compete fiercely within the first few minutes of the allocated registration time for the 'best and hot' timeslots that they wish to register. Those with high speed modems are able to key in their entries faster and stand a better chance of getting their desired timeslots. Undesirable anxieties and stress are built up among the students during subject registration days as many fear that they would not be able to get into the timeslots of their choice.

The National University of Singapore (NUS) uses the Centralized Online Registration System (CORS) (http://www.cors.nus.edu.sg/) to meet their unique registration requirements [56]. Students are allocated points to bid for their subjects. Students' choices are subsequently balloted according to the amount of the bids placed on each choice. The third local university, Singapore Management University (SMU), employs a similar bidding registration system (https://oasis.smu.edu.sg/stuportal/home/home.htm) [57]. The bidding

system allows students to rank the subjects in order of importance and essential to their respective majors. Since each subject is treated separately without considering subjects that have been registered earlier or yet to be registered, it lacks the ability to guarantee an optimal combination of the subjects that are read by a student.

Against these backdrops and also as part of a larger University Timetabling System [14], the Evolutionary Algorithm Based Subject Allocation system [45] is proposed with the intention of combining the best features of both STARS and CORS.

## 8.1. Existing Student Registration System

Currently, students in EEE are required to go through a tedious process of subject registration. They are required to plan ahead and decide which lecture group they intend to enroll in for their lecture classes. Then, they have to choose tutorial and practical classes that correspond with their choice of lecture classes. This is done by choosing an appropriate subject index number for every subject that a student wishes to enroll in. Each index number, which encodes a set of scheduled classes, is limited in its seating capacity. As in any scheduling, some of these index numbers are very popular with students while others are unpopular because they fall on very undesirable timeslots.

It is important that the set of index numbers do not result in timeslot clashes as students cannot choose two or more classes in the same timeslot. This is considered as a hard constraint, which cannot be violated.

A student will choose his/her set of index numbers that correspond with his/her personal set of preferences. Some of these preferences are: a consecutive row of classes either in the morning or afternoon, a good timeslot for lunch, a free day and avoiding classes that start very early in the morning or late in the evening. Some of these soft constraints can be violated. However, the degree of violation is used as a measure of the fitness of a particular allocation.

Students must register their set of subject index numbers within the allocated registration period. As this is an online real time system, vacancies for some index numbers are taken up very quickly. In fact, some of the vacancies are taken up within the first 30 seconds. This results in many students not having their preferred choice and they have to make quick adjustments to their pre-planned personal timetable, within the allocated registration period. Many students tend to be unhappy with this situation. Some of them, although they have registered all their subjects, may have violated many of their personal preferences (soft constraint violations). Others may have been unable to register one or more subjects because of timeslot clashes (hard constraint violations). These students will then seek the assistance of

the administrator to resolve their timetables. Hence, there is a real need to design a system that can allocate as many as possible, the students' choices in a fair and efficient manner.

EEE subject allocation system is very similar to the well-known timetabling problem. It is a multitude of many individual timetables, one for each student. As in a timetabling problem, there are limited resources (timeslots) to be shared by a pool of requests (student choices). There are also considerations of soft and hard constraint violations, which are important issues in a timetabling problem.

## 8.2. An Evolutionary Based Subject Allocation System (SALS)

The proposed SALS adopts a modified version of the conventional EA. It requires the students to provide two sets of index numbers (first and second choice). The input module of the system ensures that there are no timeslot clashes (hard constraints) within each set of index numbers. Typically, students enter 8 to 9 index numbers per set. Hence, violation of any hard constraints is prevented at the data entry level.

The index numbers within a set is a representation of a student's choice of subjects and preferences. The proposed system will try to satisfy as many students as possible by trying to allocate either their first or second choice. At any time, if there is no more vacancy for a particular

index number, the whole set in which this index number belongs will be rejected and all their previous allocations will be restored. In the event that a student is not allocated any of his/her choices, he/she will have to go through a second round of registration. Experience has shown that when the system tries to generate alternative timetables for the students who were not allocated any of his choices, most of the students do not like them. As such, the proposed system will not allocate any alternatives other than the student's first or second choice.

In addition to allocating as many students (either first or second choices) as possible, the system prefers allocations that have more first choices, given the same number of successful allocation. This is achieved through a carefully designed objective function, which is described below.

SALS uses the students' choices as input and builds a population of chromosomes out of these choices. It then evolves through the use of the Directed Window Mutation operator, developed earlier in this thesis.

## 8.3. Algorithm

Figure 28 depicts the algorithm of the SALS. The algorithm starts with building an initial population. This is done through randomly allocating students' choice of index numbers such that there is no violation of any hard constraints (vacancy for every index number allocated).

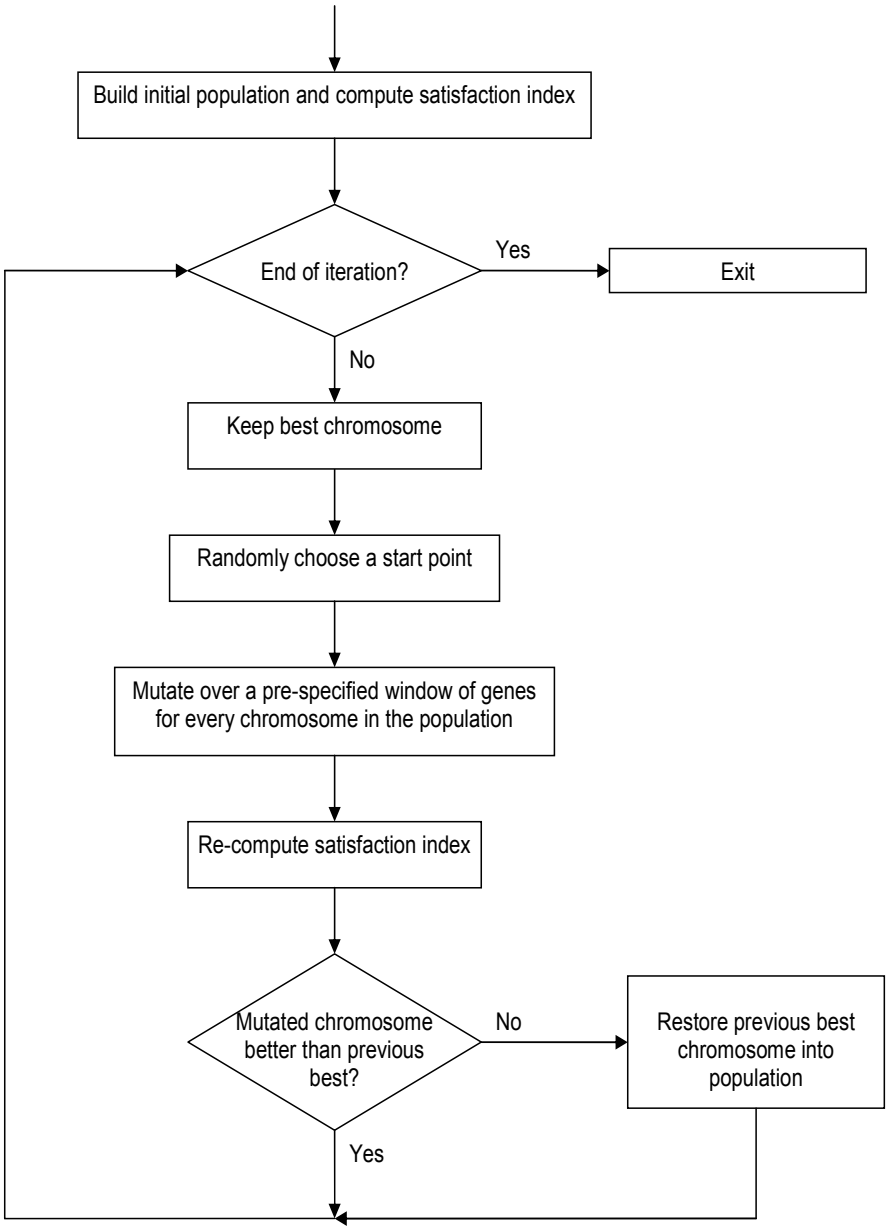



Figure 28. EA for Subject Allocation System

The system would then undergo many iterations of mutation. At the beginning of iteration, the best chromosome of the population, in terms of satisfaction index, is kept. A chromosome in the population is then picked for mutation. A start point is randomly selected over the length of the chromosome. The end point is computed by adding the mutation

window size to the start point. All the previously allocated choices between the start and end point of the chromosome are released. This is then followed by a random re-allocation of students' choices within this range, without violating any hard constraints. This is repeated for all the other chromosomes in the population.

When all the chromosomes in the population have completed their mutation, the satisfaction index of every mutated chromosome in the population is re-computed. If the best chromosome of the mutated population is inferior to the best chromosome of the previous population, the previous best chromosome will replace the chromosome with the worst satisfaction index of the mutated population. In this way, mutation is always directed towards maximizing the satisfaction index of the best chromosome.

## 8.4. Coding

Figure 29 depicts the direct coding scheme adopted by this system. Each student will have his/her own timetable. This is usually expressed as a consecutive row of index numbers. These index numbers are timeslots for tutorial and practical classes. A student has to select two sets of index numbers, which are ranked as first and second choice.

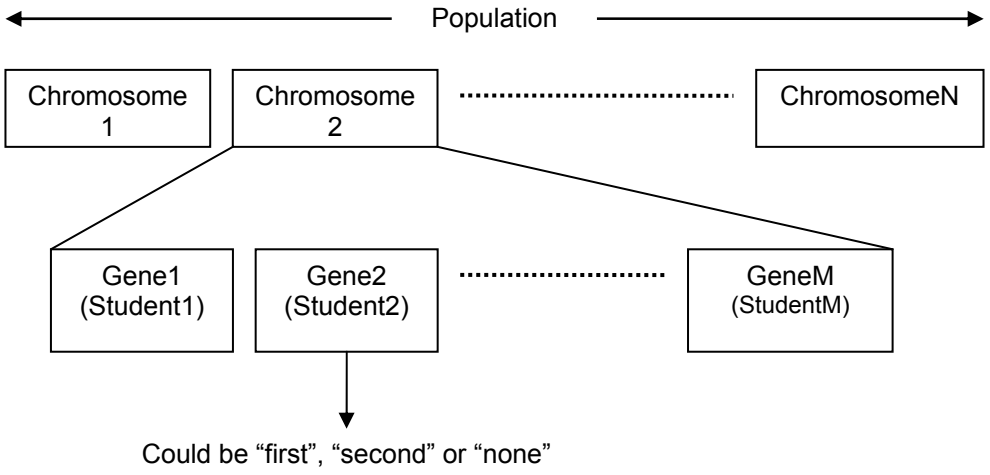



Figure 29. Coding structure of a population

A gene contains the choice allocated by the system to a student. It could appear as “first”, “second” or “none” and is defined as his/her first choice, second choice or that none of his/her choices has been allocated respectively. The number of index numbers picked by the student determines the size of the set of index numbers. Typically, it is either 8 or 9 index numbers long.

A chromosome is a string of genes. As each gene represents a possible allocation of a student, thus the number of students per selection in a cohort will determine the length of the chromosome. The order of the gene is important as it correlates with the way the students are ordered. Hence, a chromosome is a fixed length permutation of choices. DWM is used as the only evolutionary operator and as has been mentioned earlier, it is sufficient for a permutation problem such as this.

When creating an initial population, the order in which these genes are filled is done randomly. This is to remove any biasness towards the order by which students enter the system. A population consists of a set of chromosomes. The size of the population is determined empirically.

## 8.5. Fitness Computation

In timetabling scheduling, hard constraints cannot be violated. The onus is on students to enter choices that will not result in timeslot clashes. He/she has to ensure that he/she is able to attend all the classes in the two sets of subjects. The system will only ensure that the number allocated for a particular class does not exceed the seating capacity of the class.

Soft constraints are preferences that are used to compute the fitness function. Satisfaction of hard constraints is left to the students to set when they plan their sets of index numbers. In this system, the top priority is to allocate as many students as possible according to their choices. Secondly, the system will attempt to allocate more of the first choices whenever possible.

The present implementation of the system does not attempt to generate an alternative schedule if a schedule based on a student's choice is not realizable. In view of this consideration, the fitness function is formulated

based on the total number of first and second choices that are successfully allocated. In addition, it puts heavier emphasis on an allocation that satisfies more first choices. Hence, the fitness function evaluates a chromosome based on the satisfaction index according to students' choices, especially their first choices.

Let the fitness function *f* be the satisfaction index, *m* be the number of students being allocated the first choices and *n* the number of students being allocated the second choices. The fitness function *f* has two components. The first component is the ratio of the number of successful allocations (both first and second choices added together) to the total number of students. The second component gives an added weight to chromosomes having more first choices even though the number of successful allocations is the same. It is the ratio of the number of first choices to the number of successful allocations.

Mathematically, the fitness function can be written as follow:

$$f = [(m + n) / T + w\, m / (m + n)]\, 100$$

*T* is the total number of students and *w* is the weight of the second component. The value of *w* is determined in such a way that a chromosome with a greater number of successful allocations ($m + n$) has a higher fitness value as compared to another that may have more

first choices (higher *m*) but have a lower number of successful allocations (lower *m* + *n*). The scaling factor 100 is added to facilitate plotting of the results.

## 8.6. Results and discussions

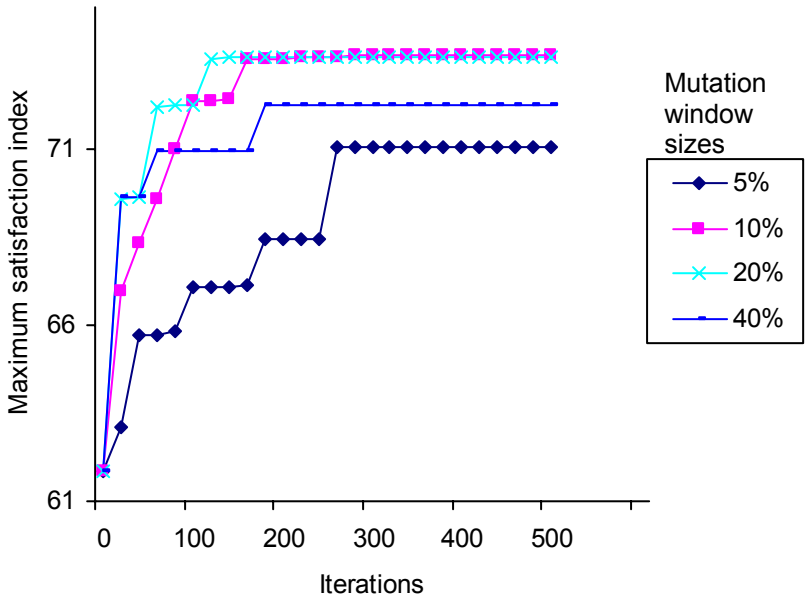



Figure 30. Maximum satisfaction index against iterations (population size = 20, different mutation window sizes)

Figure 30 presents the results obtained through testing done during semester 2 of Academic Year 2003/04. A special group of students, 77 in total, known as the Accelerated Bachelor Program (ABP) students, entered their first and second choices for a pilot test. Based on this group of students, the length of a chromosome is 77 genes. It takes about 1.5 hour to complete each simulation.

The figure shows the maximum satisfaction index plotted against iterations for different mutation window sizes. The mutation window size is measured as a certain percentage over the length of the whole chromosome. The maximum satisfaction index is the satisfaction index of the chromosome that exhibits the greatest satisfaction index of the current population.

From Figure 30, it can be seen generally that a bigger mutation window will result in better satisfaction in the long run. In the early stages of evolution, a population with a bigger window size shows greater improvement. In other words, there is a greater amount of genes participating in mutation. This would reduce the chance of being trapped in a local optimal point and improves the chance of finding another better solution. However, when the solution is close to its optimal value, a bigger window size, which means more frequent mutation, will cause greater perturbation of the population. This will result in thrashing around the optimal point, making the possibility of reaching a local peak more difficult.

A smaller window size usually takes a longer time to reach its optimal point. This is because lesser genes are involved in mutation a smaller window. As can be seen in the 5% mutation window size in Figure 30, the system takes a larger iteration number to reach an optimal point. The key reason is that less diversity is being introduced during each run.

In other words, the system may be trapped easily in a local optimal point. From Figure 31, there is a general improvement when the simulation reaches 450. This is the point where the system had nearly reached the final optimal point, where significant improvement may not be observed.

From the different simulated runs, the window size should fall between 10% and 15% at the early stage. Towards the end of evolution, a window size of 5% and 10% would be appropriate. This gives a good trade-off of diversity in the beginning and stability towards the end of the evolution process.

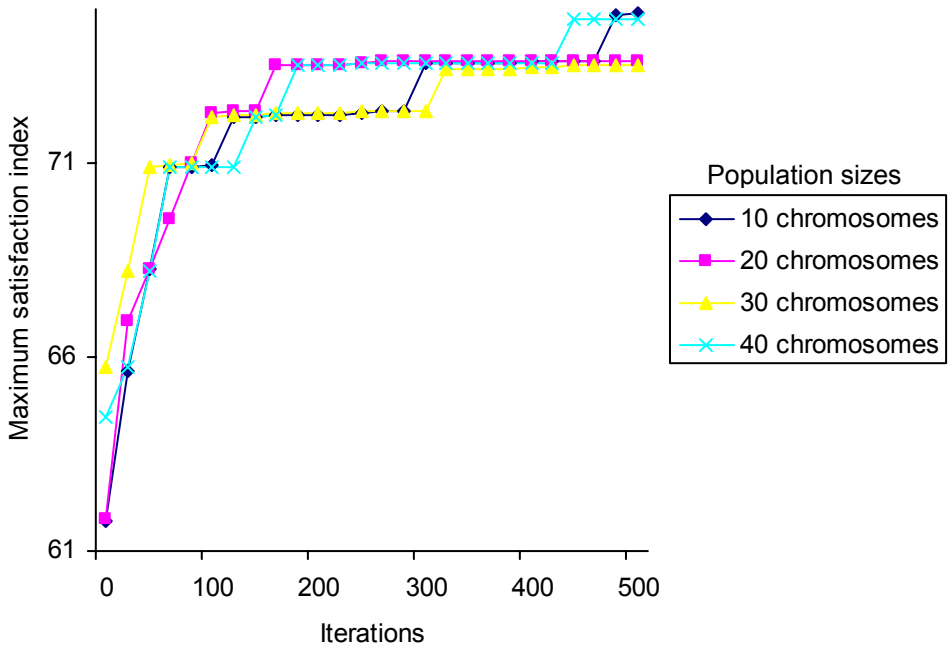



Figure 31. Maximum satisfaction index against iterations (different population sizes, mutation window size = 10 %)

As can be seen in Figure 31, the size of a population is not as important as the mutation window size. In fact, a bigger population is equivalent to more or less a smaller population with more iteration. The only advantage is that it speeds up improvement in the early stages as a bigger population implies that there is a greater chance of finding an optimal solution with less iteration. This advantage will diminish towards the end when the population reaches its optimal point. A bigger population also involves longer processing time per run. A good trade-off figure for this system is a population size of 20 to 30 chromosomes.

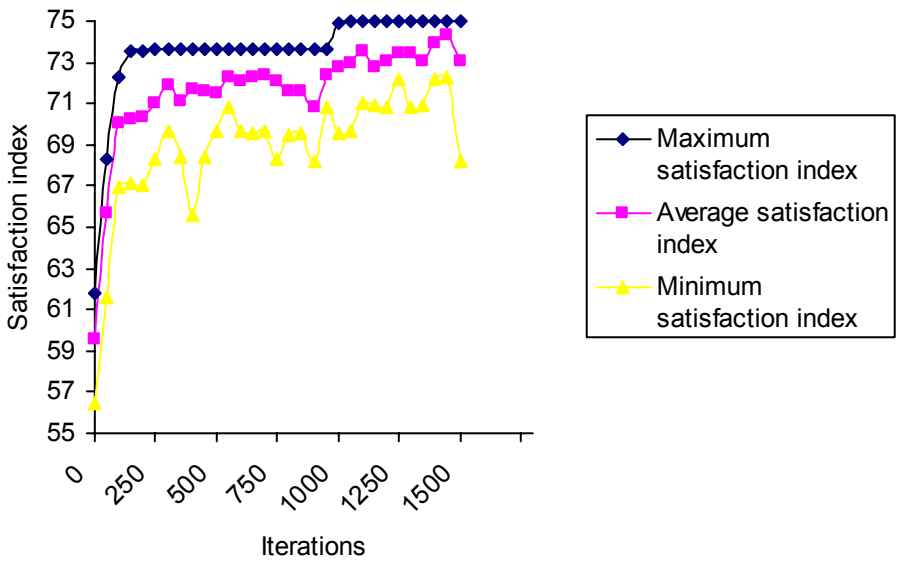



Figure 32. Maximum, average and minimum satisfaction index against iterations (population size = 20, mutation window size = 10%)

Figure 32 shows the maximum, average and minimum satisfaction index for 1500 iterations. The population size is 20 chromosomes and the mutation window size is 10% over the length of a chromosome. The value of maximum satisfaction index, which is the satisfaction index of the best chromosome in the population, rises rapidly initially and saturates towards the end. This is because the best from the previous population is either restored or superseded by better ones.

A minimum satisfaction index is attributed to the worst chromosome in the population. As can be seen in the figure, it fluctuates greatly because of mutation. However, its dip is controlled by occasionally replacing the worst chromosome with the best chromosome from the previous population. The fluctuation does provide some degrees of diversity. This could help the system to escape from a local optimal point. The distance between the maximum and minimum satisfaction index, indicates that there is a spread of satisfaction index among the population.

Table 9 shows that the number of students allocated first or second choice rose from 47 to 57 out of 77 students after 1500 iterations. This represents an improvement from 61.04% to 74.03%. Not only has the number of first and second choices increased but the ratio of the number of first choices to the number of second choices has also been

raised from 1.6 to 2.8. Although the results showed only 1500 iterations, the system is allowed to mutate up to 2000 iterations. However, no further improvement was observed.

The results are generated based on the data set with several of its index numbers heavily oversubscribed (about 3 to 4 times its seating capacity). It is believed that if some level of restriction is imposed, (for example, no index number is allowed to be oversubscribed by 2 times during data entries), the results are likely to be better. This will be one of the areas that will be explored in the future. The ultimate goal is to achieve 100% satisfaction index of either the first or second choices without being overly restrictive.

Table 9. Comparison of allocations between initial population and final population after 1500 iterations

| | Initial population | Final population |
|---|---|---|
| % Allocated | 61.04 % | 74.03 % |
| # Allocated | 47 | 57 |
| # Unallocated | 30 | 20 |
| # First Choices | 29 | 42 |
| # Second Choices | 18 | 15 |
| Satisfaction Index | 61.83 | 74.97 |

## 8.7. Conclusions

Results of testing on a sample student group have been positive. This system greatly reduces the frustration and unhappiness of students during the subject registration. In addition, it will result in significant cost and time saving for the School's Administrative body. The feedback of

this proposed system has been positive from both the students and administrators.

Initial testing based on this group resulted in limited success due to unforeseen software complication. Approximately 64% of the students were allocated schedules according to their first and second choices. The system has since been modified and as shown in Table 9 above, the satisfaction index for this data set has been further improved to 74.03%.

Another pilot run for a selected group of third year students (about 90) for the Academic Year 2004/05 Semester 1 has just been completed. In this run, the size of the mutation window was allowed to reduce with iterations. It was found that this produced better results in term of satisfaction index and the convergence rate. The success rate was 69%. In addition, partial allocation was incorporated for those students who had failed to obtain their first or second choice. In the end, every student was able to have some of his subjects registered. As this is performed upon completion of the evolutionary process, the satisfaction index of the final allocation is not affected.

The next step is to test the system with the timetable for next year and a bigger group of students (perhaps one whole academic year of 900 students). Eventually, it will be extended to the whole school (a student population of more than 3000 students). This will have a positive impact

on the subject registration system of NTU, i.e., reducing the frustration and unhappiness of the students, and saving the School Administration a great deal of time and effort.

# Chapter 9 Conclusions and Recommendations

## 9.1. Parameters used in the Proposed System

The values of the parameters, like population size, number of iterations and mutation window size, used in all the simulations are based on the finding in Section 8.8. The main reason is that the simulation in Chapter 8 is a smaller subset of the other problems. As such, it incurs less computational time to complete one simulation. However, their effects on the algorithm should not change when applied to the other two simulations.

## 9.2. Computational Time

All the simulations are done using a Pentium 3 computer with Window 98 as its operating system. Visual Basic is chosen as the language for the implementation of the proposed system. Computational times for the three simulations using the proposed algorithm show improvement as compared to existing manual methods. Existing manual method in planning a timetable would easily take a week or more. As this project is at a developing stage, the computational time includes both the user interaction time as well as the actual processing time. It took the proposed system about a day to schedule a curriculum timetable, a few hours (about 4) for examination timetabling and 1 to 2 hours for the registration of subjects. For every simulation, the processing time for a

single iteration is nearly constant. As such the results are presented using iterations rather than the actual computational time.

## 9.3. Conclusions

The main objective of this thesis work is to seek a solution to the EEE curriculum timetabling problem. At the start, this seemed to be a very challenging task as the EEE curriculum timetable is very complex and there are not many published works that deal effectively with this problem. Most of the other published works that relate to timetabling handle only the examination timetabling problem. These factors necessitate a fresh approach.

In order to formulate a good solution to the curriculum timetabling problem, the problem has to be understood and its complexity properly dealt with. The author also spent considerable time studying existing techniques that relate to the timetabling problem. A consensus agreement among the research community is to adopt a probabilistic or stochastic search technique, in particular through the use of an evolutionary algorithm.

By examining existing curriculum timetables and manually adjusting some of the allocations, several observations are noted which will help in understanding the problem. There are two main factors that determine the level of complexity.

The first factor is the number of students attending a class. A large student population, which exceeds the seating capacity of the class venue, needs to be divided into smaller sub-groups. This is equivalent to having more classes that cannot be scheduled in the same timeslot. This is an additional hard constraint. Effectively, this reduces the feasible solution space even though the search space remains the same. A smaller feasible solution space increases the difficulty to find an optimal feasible solution.

A second factor is related to the curriculum course structure of EEE. There are many types of subjects, which are translated into many types of classes that need to be scheduled. Initial approach of treating the different types of classes as one type presents many problems. Among them is the difficulty of efficiently finding an initial population of feasible solutions. On closer examination, it was noted that the problem can be eased if each type of classes is treated as a co-species (or sub-schedule) and co-evolves with one another. It was found that the best way to separate these classes is to separate them according to their duration and not according to their subject type. More improvement can be achieved by ordering the co-evolution.

Another major proposal is to do away with the crossover operation and focus entirely on mutation. This is not entirely unthinkable as we do find such approach in some published works on examination timetabling. We also find that for lower biological organisms, which do not reproduce

themselves sexually, the process of evolution is achieved through mutation. The motivation behind this is that it is easier to design a mutation operator and more efficient to execute it. It can also be shown that a crossover operation for a timetabling problem is equivalent to a mutation operation.

The proposed algorithm only allows mutation within a sub-string on a chromosome. This sub-string is like a window over the chromosome. This window shall be named as the mutation window. The rationale of using a window is to preserve as much as possible those parts of the chromosome that are good in quality. For the fitness or quality of a chromosome is determined by the presence of good quality patterns (or desirable arrangements of classes in sub-strings). A large mutation window creates more disturbances. This is good in the beginning of evolution as it provides diversity to the population. It is like having a broad coverage across the search space. A smaller mutation window is preferred towards the end of evolution. This provides stability and acts like a local search operator, such as that of a hill-climbing operator. Therefore, a good strategy is to reduce the mutation window progressively with evolution.

A final refinement is to keep a copy of the best chromosome of the previous generation and pass it on to the present generation, it is better than the best chromosome of the present generation, after undergoing

evolution. In this way, the mutation is directed and the best so far is not lost.

In this thesis work, three timetabling cases have been studied. The EEE examination timetabling problem (Chapter 5) is used to study the shortfalls of an existing evolutionary approach. Both crossover and mutation operation were used. It was found that a lot of repair works have to be done after a crossover. At the end of the repair, the resultant chromosome resembles a mutation operation over another chromosome. A local search operator was used at the end of these two operations. This was not easy to design and increased the processing time. The lessons learned are:

i) Do away with crossover and use only mutation.
ii) Use a mutation window. By adjusting its window size, we can achieve diversity at earlier stages and the local search effect at later stages of evolution.

The EEE curriculum timetabling case allowed the author to understand the complexity of the problem. One of the main challenges was to find an initial population of feasible solutions. This was extremely difficult if the classes were treated as one. The problem was greatly eased by grouping the classes according to their duration. Co-evolving (with directed window mutation) the different types of classes was then tried. Convergence was slow and not much improvement was shown. A

further refinement was added to order the co-evolution (from longer to shorter classes). This case study highlighted many of the concepts developed in this thesis and helped the author to formulate the proposed algorithm.

The EEE subject allocation case was used to study the effect of using the directed window mutation. The multi-stage co-evolutionary process was not used as they are not necessary in this case. Two pilot runs on actual data and student groups have been tried. The results were very satisfactory and spoke well of the algorithm. A further refinement incorporated a varying mutation window size in the second run. It was found that this produced better result and provided more consistent results.

The proposed evolutionary algorithm had incorporated the varying directed mutation window and multi-stage co-evolution process developed in this thesis to solve a university timetabling problem. Depending on the complexity of the problem, the algorithm could be used in part or in full.

## 9.4. Recommendations

Solving the timetabling problem using an evolutionary approach will continue to be a popular and exciting subject [30, 37, 58, 59, 60]. One of the aims of recent works in timetabling is to develop a framework or a

common methodology for a university timetabling problem [61, 62]. This thesis has helped to fulfill this aim, especially in the case of the EEE timetabling system. This could be incorporated into the other Engineering Schools, which shared similar constraints in their respective timetabling problem. Hence, a likely future direction is towards expanding such a framework by incorporating timetables of various schools.

Another immediate step is to perform actual runs and invite feedback from the staff and students for all the different types of timetables used in EEE. As EEE is a large school and the curriculum structure is still evolving, this has to be done cautiously. There is also another timetabling problem in EEE, namely the staff workload allocation system that has yet to be implemented using this novel evolutionary algorithm.

Finally, the author believes that the algorithm developed in this thesis can be applied in all the other university timetabling cases, which is usually less complex than that of NTU-EEE. Future work would also explore the possibility of having a wider application of this research work.

# Chapter 10 Author's Publications

---

## Book Chapter

Chan C K, Gooi H B, Lim M H, 2004, "Duration-Dependent Multi-Schedule Evolutionary Curriculum Timetabling", Recent Advances in Simulated Evolution and Learning, Chapter 43, page 803-820.

## Journal

Chan C K, Gooi H B, Lim M H, 2005, "An Evolutionary Algorithm Based Subject Allocation System", The Journal of the Chinese Institute of Engineers (JCIE).

## Conference

Chan C K, Gooi H B, Lim M H, 2002, "A Co-evolutionary Algorithm Approach to a University Timetable System", 2002 Congress on Evolutionary Computation, page 1946-1951, USA.

Chan C K, Gooi H B, Lim M H, 2002, "Duration-Dependent Multi-Schedule Evolutionary Curriculum Timetabling", 4$^{th}$ Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, page 677-681.

Gooi H B, Chan C K, Lim M H, 2002, “Timetables of Large Engineering College: Design and Implementation Framework”, 4th Asia-Pacific Conference on Simulated Evolution and Learning, Singapore, page 682-686.

# Bibliography

[1] Burke E K, Causmaecker P D, Petrovic S, Berghe G V, 2002, “A Multi Criteria Meta-heuristic Approach to Nurse Rostering”, The 2002 IEEE World Congress on Computational Intelligence, page 1197-1202.

[2] Burke E K, Smith A J, 1997, "A Memetic Algorithm for the Maintenance Scheduling Problem", Proceedings of the ICONIP'97 Conference, Dunedin, New Zealand, page 469-474.

[3] Chen W, Eberhart, 2002, “Genetic Algorithm for Logistics Scheduling Problem”, The 2002 IEEE World Congress on Computational Intelligence, page 512-516.

[4] Erben W, Keppler J, 1995, “A Genetic Algorithm Solving a Weekly Course-Timetabling Problem”, The Practice and Theory of Automated Timetabling: 1st International Conference, page 198-211.

[5] Meisels A, Lusternik N, 1997, “Experiments on Networks of Employee Timetabling Problem”, Lecture Notes in Computer Science 1408, Selected Papers from the Second International

Conference on Practice and Theory of Automated Timetabling II, page 130-141.

[6] Deris S, Ohta H, 1989, “A machine-scheduling model for large-scale rice production in Malaysia”, J. of the Operational Research Society, Vol. 41, No. 8, pp. 713-723.

[7] Deris S, Omatu S, Ohta H, Samat P A, 1997, “Objected-Oriented Constraint Logic Programming for Timetable Planning”, International J. of Systems Science, Vol. 28, No. 10, pp. 89-101.

[8] Deris S, Omatu S, Ohta H, Samat P A, 1997, “University Timetable Planning by Constraint-Based Reasoning – A Case Study”, J. of the Operational Research Society, Vol. 48, No. 12, pp. 1178-1190.

[9] Deris S, Omatu S, Ohta H, Samat P A, 1999, “Incorporating Constraint Programming in Genetic Algorithm for University Timetable Planning”, Engineering Applications of Artificial Intelligence, Vol. 12, No. 2, pp. 175-184.

[10] Deris S, Omatu S, Ohta H, Kutar S, Samat P A, 1999, “Ship Maintenance Scheduling by Genetic and Constraint-Based Reasoning”, European J. of Operational Research, Vol. 39, No. 2, pp. 489-502.

[11] Werra D, 1995, "Some Combinatorial Models for Course Scheduling", The Practice and Theory of Automated Timetabling: 1st International Conference, page 296-308.

[12] Wren A, "Scheduling, timetabling and rostering – a special relationship?" 1996, The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference, Lecture Notes in Computer Science 1153, pages 46-75, Springer-Verlag, berlin.

[13] Carter M W, 2000, "A Comprehensive Course Timetabling and Student Scheduling System at the University of Waterloo", Lecture Notes in Computer Science 2079, Selected Papers from Third International Conference on Practice and Theory of Automated Timetabling III, page 64-82.

[14] Gooi H B, Chan C K, Lim M H, 2002, "Timetables of Large Engineering College: Design and Implementation Framework", 4th Asia-Pacific Conference on Simulated Evolution and Learning, SEAL'02, page 682-686, Singapore.

[15] McCollum B, 1997, "The Implementation of a Central Timetabling System in a Large British Civic University", Lecture Notes in Computer Science 1408, Selected Papers from the Second

International Conference on Practice and Theory of Automated Timetabling II, page 237-253.

[16] Schaerf A, 1996, “Tabu Search Techniques for Large High-School Timetabling Problems”, proceeding of the 13th National Conference of the American Association for Artificial Intelligence.

[17] Corne D, Ross P, Fang H L, 1995, “Evolving Timetables”, Practical Handbook of Genetic Algorithms: Applications, Vol I, page 219-276.

[18] Green C D, 1999, “The Generalization and Solving of Timetabling Scheduling Problems”, Practical Handbook of Genetic Algorithms: Complex Coding Systems Vol III, page 17-63.

[19] Cooper T B, Kingston J H, 1995, “The Complexity of Timetable Construction Problems”, The Practice and Theory of Automated Timetabling: 1st International Conference, page 283-295.

[20] Garey M R, Johnson D S, 1979, “Computers and Intractability: A Guide to the Theory of NP-Completeness”, W. H. Freeman, New York.

[21] Cater M W, Laporte G, 1995 “Recent Developments in Practical Examination Timetabling”, First International Conference in Practice and Theory of Automated Timetabling, pages 3- 21.

[22] Cater M W, Laporte G, 1997, “Recent Developments in Practical Examination Timetabling”, Second International Conference in Practice and Theory of Automated Timetabling II, pages 3-19.

[23] Varter M W, Laporte G, Lee S Y, 1995, “Examination timetabling: Algorithmic strategies and application“, Dept. Industrial Eng, Univ. Toronto, Working Paper 94-03.

[24] Back T, Fogel D B and Michalewicz Z, 2000, “Evolutionary Computation Volume I: Basic Algorithms and Operators”, Institute Of Physics (IOP) Publishing.

[25] Back T, Fogel D B and Michalewicz Z, 2000, “Evolutionary Computation Volume II: Advanced Algorithms and Operators”, Institute Of Physics (IOP) Publishing.

[26] Back T, 1996, “Evolutionary Algorithms in Theory and practice”, Oxford University Press.

[27] Goldberg D E, 1989, “Genetic Algorithms in Search, Optimization and Machine Learning”, Addison-Wesley.

[28] Dawkins R, 1976, “The Selfish Gene”, London, U.K.: Oxford Univ. Press.

[29] Burke E K, Newall J P, 2003, ”Enhancing Timetable Solutions with Local Search Methods”, Lecture Notes in Computer Science, Selected Papers from Practice and Theory of Automated Timetabling IV, page 195-206.

[30] Carrasco M P, Pato M V, 2001, “A Multi-objective Genetic Algorithm for the Class/Teacher Timetabling Problem”, Lecture Notes in Computer Science 2079, Selected Papers from Third International Conference on Practice and Theory of Automated Timetabling III, page 3-17.

[31] Rudova H, Murray K, 2003, ”University Course timetabling with Soft Constraints”, Lecture Notes in Computer Science, Selected Papers from Practice and Theory of Automated Timetabling IV, page 310-328.

[32] Srinivasan D, Seow T H, Xu J X, 2002, “Constraint-Based Time-tabling Using Evolutionary Algorithm”, 4th Asia-Pacific Conference on Simulated Evolution and Learning, paper #2228.

[33] Srinivasan D, Tian H S, Jian X X, 2002, “Automated Timetable Generation Using Multiple Context Reasoning for University Modules”, 2002 Congress on Evolutionary Computation, pages 1751-1756.

[34] Ueda H, Ouchi D, Takahashi, 2001, “A Co-evolving Timeslot/Room Assignment Genetic Algorithm”, Lecture Notes in Computer Science 2079, Selected Papers from Third International Conference on Practice and Theory of Automated Timetabling III, page 48-63.

[35] Eikelder H M M, Willemen R J, 2001, “Some Complexity Aspects f Secondary School Timetabling Problems”, Lecture Notes in Computer Science 2079, Selected Papers from Third International Conference on Practice and Theory of Automated Timetabling III, page 18-27.

[36] Burke E K , Elliman D G, Weare R F, 1995, "A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems", proceedings of the 6th International Conference on Genetic Algorithms page 605-610, Morgan Kaufmann, San Francisco, CA, USA.

[37] Koizumi N, Yanamori K, Yoshihara I, 2002, "Genetic Algorithm Approach for University Timetabling", 4th Asia-Pacific Conference on Simulated Evolution and Learning, paper #1341.

[38] Burke E K, Elliman D G, Weare R F, 1993 "Automated Scheduling of University Exams", proceedings of the IEEE Colloquium on Resource Scheduling for Large Scale Planning System.

[39] Burke E K, Newall J P, 1999 "A Multi-Stage Evolutionary Algorithm for the timetable Problem", IEEE Transactions on Evolutionary Computation, Vol. 3, No 1, page.63-74.

[40] Burke E, Petrovic S, Qu R, 2002, "Case Based Heuristic Selection for Examination Timetabling", 4th Asia-Pacific Conference on Simulated Evolution and Learning, paper #2065.

[41] Ross P, Hart E, Corne D, 1997, "Some Observations about GA-based Exam Timetabling", Lecture Notes in Computer Science 1408, Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II, page 115-129.

[42] Ross P, Corne D, Fang H L, 1994, "Improving evolutionary timetabling with delta evaluation and directed mutation", Parallel Problem Solving in Nature, Volume III.

[43] Chan C K, Gooi H B, Lim M H, 2002, “A Co-evolutionary Algorithm Approach to a University Timetable System”, 2002 Congress on Evolutionary Computation, page 1946-1951, USA.

[44] Chan C K, Gooi H B, Lim M H, 2002, “Duration-Dependent Multi-Schedule Evolutionary Curriculum Timetabling”, 4th Asia-Pacific Conference on Simulated Evolution and Learning, SEAL’02, page 677-681, Singapore.

[45] Chan C K, Gooi H B, Lim M H, 2005, “An Evolutionary Algorithm Based Subject Allocation System”, The Journal of the Chinese Institute of Engineers (JCIE).

[46] Chan C K, Gooi H B, Lim M H, 2004, “Duration-Dependent Multi-Schedule Evolutionary Curriculum Timetabling”, Recent Advances in Simulated Evolution and Learning, chapter 43.

[47] Paredis J, 2000, “Co-evolutionary algorithms”, Evolutionary Computation 2, Advanced Algorithms and Operators, page 224-238.

[48] Robert V, Hertz A, 1995 “ How to decompose Constrained Course Scheduling Problems Into Easier Assignment Type Subproblems”,

The Practice and Theory of Automated Timetabling: 1st International Conference, page 364-373.

[49] Ross P, Corne D, Tersahima-Marin H, 1995, "The Phase-Transition Niche for Evolutionary Algorithms in Timetabling", The Practice and Theory of Automated Timetabling: 1st International Conference, page 309-324.

[50] Cowling P, Kendall G, Burke E K, 2002, "An Investigation of a Hyper-heuristic Genetic Algorithm Applied to a Trainer Scheduling Problem", The 2002 IEEE World Congress on Computational Intelligence, page 1185-1190.

[51] Rossi-Doria O, Sampels M, Birattari M, Chiarandini M, Dorigo M, Gambardella L M, Knowles J, Manfrin M, Mastrolilli M, Paechter B, Paquete L, Stutzle T, 2003, "A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem", Lecture Notes in Computer Science, Selected Papers from Practice and Theory of Automated Timetabling IV, page 329-351.

[52] Choong K S, Lim A S, 2000, "A Timetabling Scheduling System for the School of EEE", Final Year Project Report, Nanyang Technological University.

[53] Ross P, Corne D, Tersahima-Marin H, 1995, "The Phase-Transition Niche for Evolutionary Algorithms in Timetabling", The Practice and Theory of Automated Timetabling: 1st International Conference, page 309-324.

[54] Koh K M, 2002, "Counting", World Scientific.

[55] Deb K, 2000, "Encoding and Decoding Function", Evolutionary Computation 2, Advanced Algorithms and Operators, page 4-11.

[56] Online, available at http://www.cors.nus.edu.sg/ .

[57] Online, available at https://oasis.smu.edu.sg/stuportal/home/home.htm .

[58] Corne D, Ogden J, 1997, "Evolutionary Optimization of Methodist Preaching Timetable", Lecture Notes in Computer Science 1408, Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II, page 142-155.

[59] Paechter B, Rankin R C, Cumming A, 1997, "Improving a Lecture Timetabling System for University-wide Use", Lecture Notes in Computer Science 1408, Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II, page 156-165.

[60] Rich D C, 1995 “A Smart Genetic Algorithm for University Timetabling”, The Practice and Theory of Automated Timetabling: 1st International Conference, page 181-197.

[61] Burke E K, Kingston J H, Pepper P A, 1997, “A Standard Data Format for Timetabling Instances”, Lecture Notes in Computer Science 1408, Selected Papers from the Second International Conference on Practice and Theory of Automated Timetabling II, page 213-222.

[62] Grobner M, Wilke P, Buttcher S, 2003, “A Standard Framework for Timetabling Problem”, Lecture Notes in Computer Science, Selected Papers from Practice and Theory of Automated Timetabling IV, page 24-38.

# Appendix A Sample Input and Output

The following input and output files were used in the simulation of the 2001/2002 curriculum timetable used in Section 7.9.

## A.1. Input

Population Size = 30

Window size = 5%

Number of generations = 1650

### A.1.1. Classes

| name | code | type | group | subgroup | size | duration |
|---|---|---|---|---|---|---|
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS18 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SC | SC | 312 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS36 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS37 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS38 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS13 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS14 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS15 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS34 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS17 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS33 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS19 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS20 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS21 | 26 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS22 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS23 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS24 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SA | SA | 312 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SB | SB | 364 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS16 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS12 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS01 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS02 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS03 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS04 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS05 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS06 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS07 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS35 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS11 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS10 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS25 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SB | TS26 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS27 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS28 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS29 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS30 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS31 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SC | TS32 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS09 | 26 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SC | SC | 312 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SA | SA | 312 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SB | SB | 364 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ANALOGUE ELECTRONICS | E202 | LEC | SC | SC | 312 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SA | SA | 312 | 1 |
| ANALOGUE ELECTRONICS | E202 | LEC | SB | SB | 364 | 1 |
| ANALOGUE ELECTRONICS | E202 | TUT | SA | TS08 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS02 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS10 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS04 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS06 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS07 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS08 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS09 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS22 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS11 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS30 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS28 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS01 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS29 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS03 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS32 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS12 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS25 | 26 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS26 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS27 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS31 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS34 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS21 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS20 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS19 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS18 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS17 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS16 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS15 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS14 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS13 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS38 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS37 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SA | TS05 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS35 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS33 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SC | SC | 312 | 1 |
| ELECT MATERIALS & | E203 | LEC | SC | SC | 312 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| DEVICES | | | | | | |
| ELECT MATERIALS & DEVICES | E203 | LEC | SB | SB | 364 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SA | SA | 312 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SC | SC | 312 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SC | TS36 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SA | SA | 312 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS23 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SB | SB | 364 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SA | SA | 312 | 1 |
| ELECT MATERIALS & DEVICES | E203 | TUT | SB | TS24 | 26 | 1 |
| ELECT MATERIALS & DEVICES | E203 | LEC | SB | SB | 364 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS29 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS32 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS31 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS25 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS30 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS33 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS28 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS27 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS26 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS12 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS11 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS10 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS09 | 26 | 1 |
| NETWORK | E201 | TUT | SA | TS08 | 26 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ANALYSIS | | | | | | |
| NETWORK ANALYSIS | E201 | TUT | SA | TS07 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS06 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS05 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS34 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS03 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS17 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS02 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS04 | 26 | 1 |
| NETWORK ANALYSIS | E201 | LEC | SB | SB | 364 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS15 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS35 | 26 | 1 |
| NETWORK ANALYSIS | E201 | LEC | SA | SA | 312 | 1 |
| NETWORK ANALYSIS | E201 | LEC | SC | SC | 312 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SA | TS01 | 26 | 1 |
| NETWORK ANALYSIS | E201 | LEC | SC | SC | 312 | 1 |
| NETWORK ANALYSIS | E201 | LEC | SB | SB | 364 | 1 |
| NETWORK ANALYSIS | E201 | LEC | SA | SA | 312 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS24 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS23 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS38 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS21 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS20 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS19 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS18 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS16 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS14 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SB | TS13 | 26 | 1 |
| NETWORK ANALYSIS | E201 | TUT | SC | TS37 | 26 | 1 |
| NETWORK | E201 | TUT | SC | TS36 | 26 | 1 |

| | | | | | | |
|---|---|---|---|---|---|---|
| ANALYSIS | | | | | | |
| NETWORK ANALYSIS | E201 | TUT | SB | TS22 | 26 | 1 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS10 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS11 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS12 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS25 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS26 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS28 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS05 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS29 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS27 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS09 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS08 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS01 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS06 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS04 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS03 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS02 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS30 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS16 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SA | LS07 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS13 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS24 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS23 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS22 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS21 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS20 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS19 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS18 | 26 | 3 |
| SECOND YEAR | LAB2 | LAB2 | SB | LS14 | 26 | 3 |

| | | | | | | |
|---|---|---|---|---|---|---|
| LAB | | | | | | |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS15 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS31 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS38 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS37 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS36 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS35 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS34 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS33 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SC | LS32 | 26 | 3 |
| SECOND YEAR LAB | LAB2 | LAB2 | SB | LS17 | 26 | 3 |

## A.1.2. Resources

| Room name | capacity | Room type | special | unavailable |
|---|---|---|---|---|
| LT22 | 350 | LEC | | ,24,25,26,27, |
| LT23 | 368 | LEC | | ,24,25,26,27, |
| LT25 | 350 | LEC | | ,24,25,26,27, |
| LT27 | 430 | LEC | | ,24,25,26,27,46,47, |
| LT28 | 254 | LEC | | ,24,25,26,27, |
| LT29 | 254 | LEC | | ,24,25,26,27, |
| TR65 | 35 | TUT | | ,24,25,26,27, |
| TR66 | 35 | TUT | | ,24,25,26,27, |
| TR67 | 35 | TUT | | ,24,25,26,27, |
| TR71 | 35 | TUT | | ,24,25,26,27, |
| TR72 | 35 | TUT | | ,24,25,26,27, |
| TR73 | 35 | TUT | | ,24,25,26,27, |
| TR74 | 35 | TUT | | ,24,25,26,27, |
| TR77 | 35 | TUT | TV,VCR | ,24,25,26,27,9,18,36,45, |
| TR87 | 35 | TUT | TV,VCR | ,24,25,26,27, |
| TR102 | 37 | TUT | | ,24,25,26,27, |
| TR103 | 37 | TUT | | ,24,25,26,27, |
| TR106 | 35 | TUT | TV,VCR | ,24,25,26,27,9,18,36,45, |
| TR107 | 35 | TUT | TV,VCR | ,24,25,26,27,9,18,36,45, |
| TR108 | 35 | TUT | TV,VCR | ,24,25,26,27,9,18,36,45, |
| TR109 | 35 | TUT | | ,24,25,26,27, |
| TR110 | 35 | TUT | | ,24,25,26,27, |

| | | | | |
|---|---|---|---|---|
| TR111 | 35 | TUT | | ,24,25,26,27, |
| TR112 | 35 | TUT | | ,24,25,26,27, |
| TR113 | 35 | TUT | | ,24,25,26,27, |
| TR114 | 35 | TUT | TV,VCR | ,24,25,26,27, |
| TR115 | 35 | TUT | TV,VCR | ,24,25,26,27,9,18,36,45, |
| TR116 | 35 | TUT | | ,24,25,26,27, |
| TR117 | 35 | TUT | | ,24,25,26,27, |
| TR118 | 35 | TUT | | ,24,25,26,27, |
| TR119 | 35 | TUT | | ,24,25,26,27, |
| TR120 | 35 | TUT | | ,24,25,26,27, |
| TR121 | 35 | TUT | | ,24,25,26,27, |
| TR122 | 35 | TUT | | ,24,25,26,27, |
| LAB21 | 35 | LAB2 | | ,24,25,26,27,9,18,36,45, |
| LAB22 | 35 | LAB2 | | ,24,25,26,27,9,18,36,45, |
| LAB23 | 35 | LAB2 | | ,24,25,26,27,9,18,36,45, |
| LAB24 | 35 | LAB2 | | ,24,25,26,27,9,18,36,45, |
| LAB31 | 35 | LAB3 | | ,24,25,26,27,9,18,36,45, |
| LAB32 | 35 | LAB3 | | ,24,25,26,27,9,18,36,45, |
| LAB33 | 35 | LAB3 | | ,24,25,26,27,9,18,36,45, |
| LAB34 | 35 | LAB3 | | ,24,25,26,27,9,18,36,45, |
| PRJ21 | 35 | PRJ2 | | ,24,25,26,27,9,18,36,45, |
| PRJ22 | 35 | PRJ2 | | ,24,25,26,27,9,18,36,45, |
| PRJ23 | 35 | PRJ2 | | ,24,25,26,27,9,18,36,45, |
| PRJ24 | 35 | PRJ2 | | ,24,25,26,27,9,18,36,45, |
| PRJ31 | 35 | PRJ3 | | ,24,25,26,27,9,18,36,45, |
| PRJ32 | 35 | PRJ3 | | ,24,25,26,27,9,18,36,45, |
| PRJ33 | 35 | PRJ3 | | ,24,25,26,27,9,18,36,45, |
| PRJ34 | 35 | PRJ3 | | ,24,25,26,27,9,18,36,45, |
| DES21 | 35 | DES2 | | ,24,25,26,27,9,18,36,45, |
| DES22 | 35 | DES2 | | ,24,25,26,27,9,18,36,45, |
| DES23 | 35 | DES2 | | ,24,25,26,27,9,18,36,45, |
| DES24 | 35 | DES2 | | ,24,25,26,27,9,18,36,45, |
| DES31 | 35 | DES3 | | ,24,25,26,27,9,18,36,45, |
| DES32 | 35 | DES3 | | ,24,25,26,27,9,18,36,45, |
| DES33 | 35 | DES3 | | ,24,25,26,27,9,18,36,45, |
| DES34 | 35 | DES3 | | ,24,25,26,27,9,18,36,45, |
| DES41 | 35 | DES4 | | ,24,25,26,27,9,18,36,45, |
| DES42 | 35 | DES4 | | ,24,25,26,27,9,18,36,45, |
| DES43 | 35 | DES4 | | ,24,25,26,27,9,18,36,45, |
| DES44 | 35 | DES4 | | ,24,25,26,27,9,18,36,45, |

# A.2. Output

## A.2.1. Initial Population

**Generation 0**

**total fitness: 8931** **max fitness: 326** **min fitness: 279**
**ave fitness: 297.7**

| Pop no | badslot | conseclecture | consecvenue | goodlunch | freeday | total |
|---|---|---|---|---|---|---|
| 1 | 219 | 24 | 15 | 36 | 0 | 294 |
| 2 | 203 | 28 | 15 | 36 | 0 | 282 |
| 3 | 218 | 36 | 21 | 35 | 0 | 310 |
| 4 | 219 | 36 | 18 | 36 | 0 | 309 |
| 5 | 215 | 40 | 18 | 35 | 0 | 308 |
| 6 | 205 | 30 | 12 | 36 | 0 | 283 |
| 7 | 215 | 24 | 12 | 35 | 0 | 286 |
| 8 | 223 | 34 | 9 | 36 | 0 | 302 |
| 9 | 209 | 34 | 24 | 36 | 0 | 303 |
| 10 | 226 | 30 | 27 | 36 | 0 | 319 |
| 11 | 215 | 36 | 21 | 36 | 0 | 308 |
| 12 | 210 | 36 | 18 | 36 | 0 | 300 |
| 13 | 208 | 30 | 21 | 36 | 0 | 295 |
| 14 | 209 | 36 | 12 | 36 | 0 | 293 |
| 15 | 212 | 28 | 12 | 36 | 0 | 288 |
| 16 | 208 | 46 | 6 | 35 | 0 | 295 |
| 17 | 218 | 34 | 6 | 36 | 0 | 294 |
| 18 | 218 | 26 | 12 | 36 | 0 | 292 |
| 19 | 216 | 30 | 9 | 36 | 0 | 291 |
| 20 | 223 | 46 | 21 | 36 | 0 | 326 |
| 21 | 206 | 44 | 18 | 36 | 0 | 304 |
| 22 | 213 | 22 | 9 | 36 | 0 | 280 |
| 23 | 215 | 40 | 15 | 36 | 0 | 306 |
| 24 | 216 | 38 | 12 | 35 | 0 | 301 |
| 25 | 212 | 44 | 6 | 36 | 0 | 298 |
| 26 | 206 | 46 | 27 | 36 | 0 | 315 |
| 27 | 212 | 32 | 12 | 36 | 0 | 292 |
| 28 | 217 | 30 | 15 | 34 | 0 | 296 |
| 29 | 211 | 26 | 9 | 36 | 0 | 282 |
| 30 | 205 | 30 | 9 | 35 | 0 | 279 |

# A.2.2. Final Population

**Generation 1650**

**total fitness: 10829** **max fitness: 424** **min fitness: 321**
**ave fitness: 360.966666666667**

| Pop no | badslot | conseclecture | consecvenue | goodlunch | freeday | total |
|---|---|---|---|---|---|---|
| 1 | 245 | 48 | 33 | 36 | 0 | 362 |
| 2 | 241 | 40 | 33 | 36 | 0 | 350 |
| 3 | 240 | 36 | 21 | 36 | 0 | 333 |
| 4 | 261 | 62 | 45 | 36 | 1 | 405 |
| 5 | 265 | 66 | 54 | 36 | 3 | 424 |
| 6 | 239 | 44 | 36 | 36 | 0 | 355 |
| 7 | 247 | 56 | 45 | 36 | 1 | 385 |
| 8 | 241 | 42 | 36 | 36 | 0 | 355 |
| 9 | 245 | 34 | 30 | 36 | 0 | 345 |
| 10 | 249 | 62 | 42 | 36 | 0 | 389 |
| 11 | 264 | 64 | 51 | 36 | 1 | 416 |
| 12 | 236 | 48 | 39 | 36 | 0 | 359 |
| 13 | 237 | 40 | 30 | 36 | 0 | 343 |
| 14 | 246 | 46 | 33 | 36 | 0 | 361 |
| 15 | 253 | 64 | 48 | 36 | 0 | 401 |
| 16 | 228 | 48 | 42 | 36 | 0 | 354 |
| 17 | 237 | 48 | 33 | 36 | 0 | 354 |
| 18 | 234 | 32 | 21 | 36 | 0 | 323 |
| 19 | 232 | 48 | 36 | 36 | 0 | 352 |
| 20 | 244 | 40 | 33 | 35 | 0 | 352 |
| 21 | 242 | 38 | 27 | 36 | 0 | 343 |
| 22 | 235 | 34 | 18 | 36 | 0 | 323 |
| 23 | 233 | 34 | 30 | 36 | 0 | 333 |
| 24 | 258 | 54 | 39 | 36 | 2 | 389 |
| 25 | 244 | 34 | 30 | 36 | 0 | 344 |
| 26 | 246 | 58 | 45 | 36 | 0 | 385 |
| 27 | 243 | 40 | 36 | 36 | 1 | 356 |
| 28 | 228 | 48 | 39 | 36 | 0 | 351 |
| 29 | 224 | 34 | 27 | 36 | 0 | 321 |
| 30 | 248 | 42 | 39 | 36 | 1 | 366 |