MEMETIC ALGORITHMS FOR TIMETABLING PROBLEMS IN PRIVATE SCHOOLS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

DENİZ ALDOĞAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

JUNE 2005

Approval of the Graduate School of Natural and Applied Sciences

_____

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____	_____
Assistant Prof. Dr. Ender Özcan Co-Supervisor	Assoc. Prof. Dr. Ferda N. Alpaslan Supervisor

Examining Committee Members

Assoc. Prof. Dr. Ferda N. Alpaslan(METU,CENG) _____

Assoc. Prof. Dr. Nihal K. Çiçekli (METU,CENG) _____

Assoc. Prof. Dr. Ali Doğru (METU,CENG) _____

Dr. Ayşenur Birtürk (METU,CENG) _____

Assistant Prof. Dr. Ender Özcan (Yeditepe Uni.) _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Deniz Aldoğan

Signature :

ABSTRACT

MEMETIC ALGORITHMS FOR TIMETABLING PROBLEMS
IN PRIVATE SCHOOLS

Aldoğan, Deniz
M.S., Department of Computer Engineering
Supervisor : Associate Prof. Ferda Nur Alpaslan
Co-Supervisor : Assistant Prof. Ender Özcan

June 2005, 156 pages

The aim of this study is to introduce a real-world timetabling problem that exists in some private schools in Turkey and to solve such problem instances utilizing memetic algorithms.

Being a new type of problem and for privacy reasons, there is no real data available. Hence for benchmarking purposes, a random data generator has been implemented. Memetic algorithms (MAs) combining genetic algorithms and hill climbing are applied to solve synthetic problem instances produced by this generator.

Different types of recombination and mutation operators based on the hierarchical structure of the timetabling problem are proposed. A modified version of the violation directed hierarchical hill climbing method (VDHC), introduced by A. Alkan and E. Ozcan, coordinates the process of 12 different low-level hill climbing operators grouped in two distinct arrangements that attempt to resolve corresponding constraint violations. VDHC is an adaptive method advocating cooperation of hill climbing operators. In addition, MAs with VDHC are compared with different versions of multimeme algorithms and pure genetic algorithms.

Experimental results on synthetic benchmark data set indicate the success of the proposed MA.

Keywords: Evolutionary Computing, Memetic Algorithms, Timetabling Problems

ÖZ

ÖZEL OKULLARDAKİ ZAMAN ÇİZELGELEME PROBLEMİ İÇİN MEMETİK ALGORİTMALAR

Aldoğan, Deniz
Y. Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi : Doçent Ferda Nur Alpaslan
Ortak Tez Yöneticisi : Yardımcı Doçent Ender Özcan

Haziran 2005, 156 sayfa

Bu çalışmanın amacı, Türkiye'de bazı özel okullarda var olan gerçek bir zaman çizelgeleme problemini tanıtmak ve memetik algoritmalardan yararlanarak bu tip problem örneklerini çözmektir.

Yeni bir problem tipi olmasi ve gizlilik nedenleri dolayısıyla kullanilabilir gerçek data mevcut değildir. Bu nedenle, karşılaştırma amaçları için rastgele veri yaratan bir program gerçekleştirilmiştir. Genetik algoritmaları ve tepe-tırmanmayı birleştiren memetik algoritmalar, bu programla üretilmiş sentetik örnekleri çözmek için uygulanmıştır.

Zaman çizelgeleme probleminin hiyerarşik yapısına dayanan farklı rekombinasyon ve mutasyon operatörleri önerilmiştir. A. Alkan ve E. Ozcan tarafından tanıtılan bozulma güdümlü hierarşik tepe tırmanma yönteminin(VDHC) değişik bir versiyonu, iki farklı düzenleme ile gruplanmış, ilişkin kısıtlama bozulmalarını çözmeye çalışan 12 değişik aşağı seviye tepe tırmanma operatörlerini koordine eder. VDHC, tepe-tırmanma operatörlerinin işbirliğini koruyan uyarlanabilir bir yöntemdir. Ek olarak, VDHC ile beraber memetik algoritmalar multimeme algoritmalarin değişik versiyonlari ve saf genetik algoritmalar ile karşılaştırılmıştır.

Sentetik ölçüm verileri kümesi üzerindeki deneysel sonuçlar önerilen memetik algoritmanin başarısını göstermektedir.

Anahtar Kelimeler: Evrimsel Hesaplama, Memetik Algoritmalar, Zaman Çizelgeleme Problemleri

To Nejla, Ulaş and Şahin Aldoğan

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my supervisor, Ferda N. Alpaslan for guidance and sound advice throughout the course of my study.

I am also deeply grateful to Ender Özcan for devoting a vast amount of hours to guidance, advice, insight, discussion, always being very open-minded and equally patient.

I would like to thank to Alpay Alkan for providing me with various constraint specifications for private school timetabling.

I finally would like to thank to the administration in the Computer Engineering Department of Yeditepe University for offering the utilization of the department's labs during the experiments of this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

The aim of this thesis is to introduce a new timetabling problem, namely the timetabling problem for private schools (or test preparation schools), and to apply memetic algorithms for the solution of this problem. In addition, different crossover and mutation operators are proposed and compared with traditional genetic operators.

In the proposed framework, different low level hill climbing operators specific to the considered timetabling problem instances are developed and coordinated by a higher-order method, the VDHC method. This method has been implemented and used for several timetabling problems involving a single hierarchical organization in their structure. In this study, this method has been extended to work for a different timetabling problem that comprises two different hierarchical trees in its structure. Accordingly, the proposed low-level hill climbers belong to one of the two hierarchical organizations. Comparison of the proposed memetic algorithm with multimeme algorithms and pure genetic algorithms on synthetic benchmark data is also performed.

Chapter 1 introduces the timetabling problem. Chapter 2 presents a new class of timetabling problems. Chapter 3 summarizes previous research on high-school timetabling problems since they most resemble the new timetabling problem introduced. Chapter 4 gives an overview of evolutionary algorithms. Chapter 5 discusses the basics and possible design methods of memetic algorithms. Sample studies on timetabling problems with memetic algorithms are also mentioned in this chapter. Chapter 6 gives a brief overview of multimeme algorithms. Chapter 7

discusses the proposed memetic algorithm framework in detail. Chapter 8 explains the multimeme side of the framework. In chapter 9, the random data generator for creating synthetic data for the newly introduced timetabling problems is explained. Chapter 10 summarizes the experiments and discusses the results. Chapter 11 summarizes the work done and the conclusions drawn in this study.

1.1 The Timetabling Problem

The timetabling problem aims to achieve feasible assignments for a collection of events that are required to take place within a finite period of time such that necessary constraints are satisfied. Among all the possible constraints that may be defined for a timetabling problem instance, a very fundamental constraint requires that no two events corresponding to the same resource must be scheduled at the same time. For instance, no two different lecture hours of the same instructor can be assigned at the same time slot for a course timetabling problem. Generally, constraints for a timetabling problem can be classified as hard and soft constraints. All of the hard constraints must be satisfied for a solution instance to be feasible, i.e to be put into use, whereas soft constraints denote preferences and their violations may be tolerated to some extent. As a consequence, a high-quality solution for the timetabling problem is the one that contains no hard constraint violations and a minimum number of soft constraint violations.

The timetabling problem can be modelled in terms of various concepts such as graph theory or integer programming. In the work of Burke et. al. (1995c), it is mentioned that the problem of assigning events, i.e exams in their case, to time slots is equivalent to the problem of assigning colors to vertices in a graph. If each event in the timetabling problem is drawn as a vertex in the graph, conflicting events, i.e events that must not be scheduled at the same time, can be identified as follows: an edge is created between each pair of two vertices that denote a pair of conflicting events and vertices that have an edge between them

are assigned to different colors while coloring all the vertices. If each color represents a different period, we make sure that no conflicting events are assigned at the same time slot with this coloring scheme. Figure 1 displays a sample case, where different shapes for the events indicate different colors assigned to them. In that graph, there are 8 edges, which means there are 8 pairs of conflicting events. Time slots are assigned to events according to these edge constraints. For instance, event 1 is assigned to a different period than the periods assigned to event 2, 4 or 5.

In the study of Burke et. al. (1995c), several studies such as that of Welsh et. al. (1967) that attempt to solve the timetabling problem by means of graph coloring are referred.

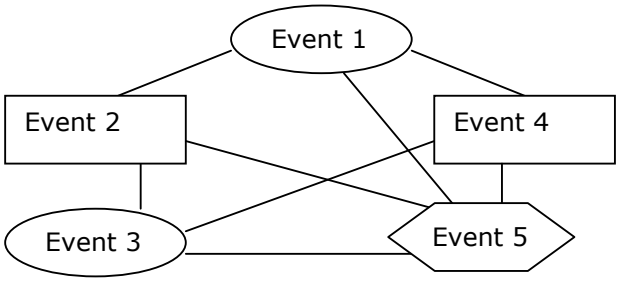

Figure 1.1 Equivalence of Timetabling Problems with Edge Constraints and Graph Coloring

Timetabling problems, which are real-world constraint optimization problems, are NP-complete problems (Even et. al. 1976). Therefore, a timetabling problem such as many optimization problems cannot be solved by optimizing the value of each variable while neglecting the others. The reason for this can be traced to the interactions between several parameters of the problem.

There are several categories for timetabling problems such as nurse rostering problems, university course timetabling problems, university final exam timetabling problems and high-school timetabling problems. In nurse rostering problems, shift assignments and rest days of the nurses must be satisfactorily placed in the timetable. In fact, nurse rostering is a subproblem of a broader range of timetabling problems, namely the employee timetabling problems. University course or exam timetabling involve scheduling a set of courses or exams given a set of constraints. High-school timetabling problems differ from these two problems since achieving a minimum number of gaps, i.e empty slots, is a main requirement for such problems. This constraint is also necessary for the newly introduced private school timetabling problem. Hence, a literature survey on methods for solving high-school timetabling problems becomes necessary in this study.

CHAPTER 2

THE TIMETABLING PROBLEM FOR PRIVATE SCHOOLS IN TURKEY

This section aims to introduce the notion of private schools in Turkey and outline the timetabling problem for them. Firstly, the need for such institutions is figured out. Then, their basic organization is described. After that, a more formal definition for the timetabling problem of private schools is discussed along with the possible constraints that may be involved in the problem.

2.1 The Need for Private Schools – OSYM

The universities, which are the principal higher education institutions in Turkey, all accept their students in accordance with the results of an examination organized by The Student Selection and Placement Center(OSYM). OSYM aims fair access to higher education programs by providing a centralized system for admission of students to the institutions of higher education. The entrance examination system is essentially based on a one-stage examination, namely the Student Selection Examination(OSS), which is held throughout the country once a year. So, every year the bulk of the students for undergraduate programs of the universities(i.e., those admitting students with a high school diploma or its equivalent) are selected and placed by this centrally administered examination system.

OSS comprises of two tests. One of them is prepared to measure mainly the candidates' verbal abilities, and the other, their quantitative abilities. After the completion of score transformations, three different composite scores are calculated for each candidate and used in selection of those candidates who will be considered for placement in the undergraduate

programs. These scores are verbal, quantitative and equally weighted OSS scores. Every department in all of the universities accepts students according to a specific type of OSS scores. Hence, each student tries to maximize one of the three types of those OSS scores to access to his/her desired department in a university. As a result, private schools that prepare students for OSS in Turkey generally assign their students to one of the three different divisions, each of which prepares a student to maximize one of his/her verbal, quantitative or equally-weighted scores. Therefore, each division of a private school is devoted to a specific score type.

OSS is not the only centralized examination held by OSYM. There are many other such examinations. In accordance, there are many private schools that prepare students to various examinations held by OSYM. Those institutions offer several programs, each of which is dedicated to a specific examination. In this study, a possible program for OSS of such an institution will be introduced for illustration purposes, but a similar description can be applied to other program instances as well.

2.2 Private Schools in Turkey

A private school has a number of instructors employed, a number of students registered and several branches, each of which are located in different buildings. Each branch of the private school is divided into different grades. A grade identifies which high school class a student attends to or whether he/she is a high school graduate. Hence, the curriculum of a second-year high school student in a private school differs from that of a third-year high school student. Those grades are further divided into divisions. For instance, a private school branch that prepares students for OSS can have three types of divisions, each of which has a curriculum to improve one of the verbal, quantitative or equally weighted OSS scores of students. Also, divisions are divided into sections(i.e classes). Students in each section attend to lectures for a number of hours on a number of days in a week. All the students in a section have

the same weekly timetable. In addition, a specific timetable for every section that the private school offers must be constructed. There are a number of courses offered to students of a private school. Courses are divided into course-sections, each of which is assigned to a specific section of a private school. Therefore, the number of course-sections of a course is equal to the number of sections whose students must attend to the course. In addition, the weekly number and length of meetings for each of those course-sections are given. Usually, the number and length of meetings for a course-section(i.e total hours for a course-section) increase with the difficulty of the course. These predefined values are generally the same among all sections of the same division since a discrimination among them is often avoided.

A section must be assigned to several course-sections according to its division. To illustrate, a section for third-year high school students that are in the quantitative division take mathematics courses, courses in natural sciences and Turkish language courses. All the students that are in the verbal or equally weighted divisions take courses in social sciences such as geography or history as well as mathemathics and Turkish language courses. However, the number and lenght of meetings that must be assigned to the sections of different divisions can differ. To illustrate, a section of the equally weighted division may have 4 meetings of Geography lectures, whereas a section of verbal division may have 6 meetings of Geography assigned.

The students of each section are registered to a specific branch and attend their courses only in that location. However, instructors may give lectures at several different branches of the institution. The organization of a private school is displayed in Figure 2.1.

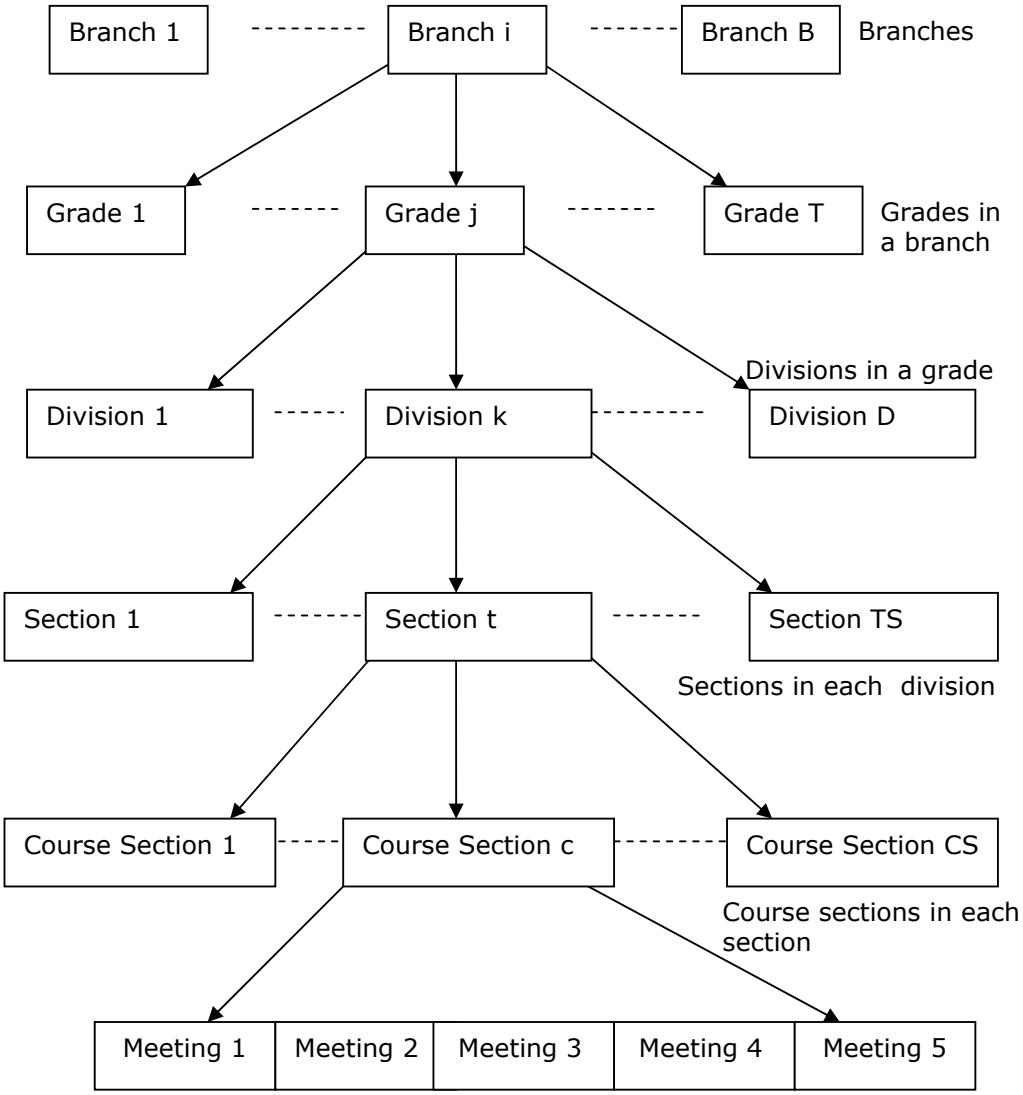

Figure 2.1 The Organization of a private school

2.3 The Private School Timetabling Problem

Generally, a timetable problem can be described as in the work of Corne (1994). There is a finite set of events E= $\{e_1, e_2, ..., e_v\}$, a finite set of potential fixed-length periods for these events T= $\{t_1, t_2, ..., t_n\}$, a finite set of places where the events can occur P= $\{p_1, p_2, ..., p_n\}$ and a finite set of agents A= $\{a_1, a_2, ..., a_m\}$ that act in a subset of the events. In the private school timetabling problem, events correspond to meetings of each course-section, periods are the available teaching hours defined by the private school, places are the classrooms belonging to a specific branch and agents are the instructors. Here, it should be noted that the set of events in private school timetabling cover all the course-sections that are offered for all the sections in the branches of the private school. Each specific course-section meeting has a length and each classroom has a capacity. In addition, travelling times between classrooms in different branches must be considered.

An assignment in the timetabling problem is an ordered 4-tuple (a,b,c,d), where a $\in$ E, b $\in$ T, c $\in$ P, d $\in$ A. The interpretation of this assignment in terms of private school timetabling problem is: "Course-section meeting a starts at time b in room c, and is taught by instructor d". So, the private school timetabling problem is to produce a feasible timetable, a collection of assignments one per course-section meeting, with minimum number of constraint violations.

Since each course section is offered for a particular section, i.e a class, we needn't produce assignments of course-sections to sections in the timetabling problem. Also, instructor assignments for each course-section meeting of all courses is usually given in advance. Therefore, the problem generally boils down to figuring out feasible time assignments for each of the events.

In order to solve the private school timetabling problem, the timetables for all the sections must be produced by assigning their course-section

meetings to time periods. The timetables for each specific section cannot be solved in isolation from the other sections. Otherwise, there is no way of controlling whether an instructor is assigned to two or more course-sections at the same time period.

2.4 Private School Constraints

The types of constraints for the timetabling problem of a private school can be listed as follows.

2.4.1 Unary Constraints

Unary constraints involve just one event and appear in the form of preset or exclusion constraints. An exclusion indicates which of the resources are unavailable for an event, whereas a preset constraint represents the predefined allocation of a resource(or resources). Constraints in this group are:

1) Meetings of course-sections can be assigned to predefined hours in predefined days(CS_PRE).
2) Meetings of course-sections should be assigned to allowable hours of the corresponding sections(S_PRE, S_EXC).

2.4.2 Binary Event Constraints

Binary constraints appear because of the restrictions involving the assignments of a pair of events. Constraints of this type can be summarized as juxtaposition or ordering constraints that restrict the order and time gap between two events. A subset of juxtaposition constraints are the edge constraints, which require that no overlapping of resources must occur for the assignments of two events. These are the most crucial constraints that contribute to the feasibility of a timetable. Such constraints are:

1) Each section is assigned to at most one course-section at a given time period(EC1).
2) Each instructor is assigned to at most one course-section at a given time period(EC2).
3) Each classroom is assigned to at most one course-section at a given time period(EC3).

2.4.3 Event-spread Constraints

Event-spread constraints restrict how events should be spread out in the timetable. Below are such constraints:

1) Each meeting of a course-section should be assigned to different days in a week (ES1).
2) An even distribution of verbal and quantitative courses should be achieved by assigning minimum and maximum hours for each course type in a day for the sections. These constraints are named as S_DIVMAXWL, daily maximum workload of all the courses from a specific division, and S_DIVMINWL, daily minimum work of all the courses from a specific division, respectively. To illustrate, these constraints can be defined for a section as follows: Students of section S_i can have maximum 6, minimum 3 hours of courses offered from the verbal division in a day.
3) There should be a minimum number of gaps between course-sections assigned to a section in a day unless stated otherwise(i.e there can be a one-hour lunch break). Hence, compactness of the daily timetable for a section should be achieved(S_MINGAP).
4) Course-sections assigned to an instructor in a day should be consecutive and should contain a minimum number of gaps. This constraint is necessary so that instructors who are paid for each lecture hour can finish their work at the private school in a shorter period. So, compactness of the instructor daily timetables can also be required(I_MINGAP).

5) Number of hours of course-sections assigned to a section in a day should be within predetermined minimum and maximum hours(S_MAXWL, S_MINWL). For example, we can define these constraints as follows: Students of section S_i can have maximum 6, minimum 3 hours of courses in a day.
6) Courses requiring more intellectual activity should be placed earlier in the timetables than the ones that are generally accepted as being rather easier. For example, courses of natural sciences should be held earlier in the morning, whereas a Turkish language course can be placed in the afternoon or evening according to the predefined available hours. We can take this constraint as an ordering constraint as well(ES2).

2.4.4 Instructor Constraints

Finally, there are some instructor constraints as described below.

1) The specifications involving allowable hours of instructors should be satisfied (I_EXC).
2) Since a private school can have several branches each in different buildings, travelling times of instructors between these buildings should be minimized. In private schools, this is generally achieved by restricting the number of different branches at which an instructor lectures in a day (I_MAXLOC).
3) Number of hours of course-sections assigned to an instructor in a day should also be within predetermined minimum and maximum hours(I_MINWL, I_MAXWL).
4) Travelling times of instructors should also be considered when assigning course-sections to them in a day. To illustrate, there should be at least one hour travelling time left between two course-sections that will be offered to sections of different branches in different locations by the same instructor(I_TRAVEL).

2.5 How Difficult is Private School Timetabling?

The factors that contribute to the difficulty of a specific timetabling problem, namely the examination timetabling, have been revealed in a survey by Burke et.al (1995b). In this survey, they intended to bring together all variations of the exam timetabling problem in British Universities, which was stated to be the first step of the unification of all timetabling problems.

They considered how several aspects such as the number and length of exams, number of students, number of departments, number and length of periods, availability of rooms and invigilators affected the difficulty of exam timetabling. In addition, they figured out the average importance of common constraints of the problem. As in any timetabling problem, the difficulty brought about by the variables boils down to the constraints defined in the private school timetabling problem. Hence, we cannot estimate the complexity of any timetabling data without regarding the constraints.

The difficulty of solving a private school timetabling data can be altered by the values and ratios of the variables in the problem. The interactions of such variables are considered by the aid of several such ratios defined in the following sections.

2.5.1 The Effect of Instructors

It becomes more difficult to achieve feasible assignments of course sections to available time periods as the ratio of average number of instructors for a course over the average number of course sections for a course decreases. The reason is that there is more possibility of assigning two course sections with the same instructor to the same time slot. In fact, the number of course sections for a specific course is exactly equal to the summation of number of sections that take the course according to the curriculum of their divisions. Hence the ratio above can also be stated

as the average number of instructors for a course over the average number of sections that take the course.

To be more specific, the total number of meetings or hours for course sections can replace the denominator of the described ratio. This is because the actual events to be assigned to time slots are course section meetings in the private school timetabling.

Another ratio involving the number of instructors can be stated as follows: the number of branches over the number of instructors. If we keep the total number of instructors constant, assignments of instructors to course sections held at different branches will be inevitable as the number of branches, therefore the number of course sections at these branches, increases. Therefore, the number of travelling instructors will be increased, which will put more burden on achieving the feasibility of the timetable while more travelling times to be considered are introduced.

In conclusion, average number of assigned time slots for an instructor gives us insight to reveal the difficulty of the current problem since there will be more event clashes due to instructor assignments if the average workload of an instructor increases. This ratio, i.e total number of course meeting hours over total number of instructors, is defined as the average occupancy for instructors in the study of Alkan et. al. (2003).

2.5.2 The Effect of Students

The number of students registered in a branch of a private school determine how many sections will be offered at that branch. As the number of sections increases, figuring out feasible assignments of course-sections for these sections to time slots will get harder since accomodation of sections will also be a problem as well.

2.5.3 The Effect of Course Section Meetings

Increasing the total number of meetings for course sections will make it more difficult to construct the timetables of corresponding sections. This is simply because the number of events to be scheduled will increase. Therefore, an increase in the ratio of number of meetings for all courses over the number of time slots will cause more difficulties in the time assignments of events.

Besides, the average number of assigned hours for a section can be used to figure out the difficulty of the problem instance since there will be more event clashes due to section assignments if the average workload of a section increases.

2.5.4 The Effect of Constraints

The density of conflicting events in the timetabling problem gives us insight to reveal the difficulty of the problem. In the private school timetabling case, the most crucial constraints that must be satisfied for the feasibility of the solution are the edge constraints. They make sure that no two course section meetings that are assigned to the same class, i.e section, or instructor should overlap. So, if two events in the private school timetabling have the same instructor assigned or if they belong to the same section, they are in fact two conflicting events.

All the events defined in a private school timetabling problem have the possibility of conflicting due to instructor assignments. For instance, a course section offered for second grade students may conflict with another course for third grade students if the same instructor is assigned to both of the course sections. Therefore, while deriving the density of conflicting events in the private school timetabling, we need to consider all the events in the problem.

Assuming there are N events, i.e course section meetings, for the problem instance, there can be at most N(N – 1)/2 conflicting events. By counting the number of conflicting course section meetings and dividing them by the total number of conflicts that may exist, we acquire edge-constraint density(Table 2.1).

Table 2.1 Calculation of Conflict Density for Edge Constraints

```
Conflict Density Calculation (ParameterSet p)
p->ConflictCount =0
//p->EventNo gives the number of course section meetings, i.e events
for i=0 to p->EventNo do
  for j=i+1 to p->EventNo do
    if  p->CourseSections[i].InstructorID==p->CourseSections[j].InstructorID
      OR
    p->CourseSections[i].SectionID==p->CourseSections[j].SectionID
    then
      p->Conflict_Count++;
    end if
  end for
end for
p->ConflictDensity = p->ConflictCount/((p->EventNo)(p->EventNo - 1)/2);
```

2.5.5 Conflict Density Analysis

Assume that there are totally N sections in a private school timetabling problem. Then, the conflict density for EC1 becomes:

$$\frac{\sum_{s=1}^{N} M_s\,(M_s - 1)(1/2)}{E\,(E - 1)(1/2)} \qquad (2.1)$$

where M_s is the number of meetings for section s and E is the total number of meetings in the problem. Furthermore, let M denote the

average number of course section meetings, i.e events, assigned to a section. The total number of events in this problem is then approximately NM. Therefore, the maximum number of edge constraints defined for sections, say EC1_MAX, can be defined as follows:

$$EC1\_MAX \approx (N\,M)(N\,M-1)(1/2) \qquad (2.2)$$

Since each section with M events introduce M(M-1)(1/2) EC1 constraints, the number of EC1 constraints will infact be approximately NM(M-1)(1/2). Hence, the conflict density of EC1 constraints will generally be approximately (M-1)/(NxM-1) which is always lower than 1/(N-1) for (N+M)>2 , since (NM-1)+2-(N+M) equals to (N-1)(M-1). As a result, for problems where section number is reasonably large, i.e N>10, conflict density of EC1 constraints will always be less than 0.1 and thus the density of those constraints are bounded.

A similar analysis can be performed for bounding EC2 constraint density. This time, assume that there are totally P instructors, where the average number of course section meetings assigned to an instructor is Q. In this case, the total number of events in this problem will be approximately PQ. Therefore, there can be at most (PQ)(PQ-1)(1/2) edge constraints defined for instructors(EC2). Since each instructor with Q events introduce Q(Q-1)(1/2) EC2 constraints, the number of EC2 constraints is infact PQ(Q-1)(1/2). Hence, the conflict density of EC2 constraints will generally be approximately (Q-1)/(PQ -1) which is always lower than 1/(P-1) for (P+Q)>2 again since (PQ-1)+2-(P+Q) equals to (P-1)(Q-1).

As a result, for problems where number of instructors is reasonably large, i.e P>10, conflict density of EC2 constraints will always be less than 0.1 and thus the density of those constraints are bounded. To conclude, the conflict density due to edge constraints in a private school problem instance will generally be less than 0.2. However, this does not indicate that private school problems are rather trivial timetabling problems since

many other constraints contribute to the difficulty of the problem such as minimum gap or workload constraints.

CHAPTER 3

LITERATURE REVIEW FOR HIGH SCHOOL TIMETABLING

This thesis aims to solve the timetabling problem for a private school. The nature of such a problem resembles the high school timetabling problem more than university or exam timetabling problem due to several reasons. Firstly, both private schools and high school students are grouped in classes. Generally, these groups of students are disjoint. Each student belonging to a class takes the courses that are listed in the fixed curriculum of this class. Therefore, these students do not have the freedom for selecting most of their courses, whereas university students do. Secondly, compactness of the resulting timetable for each class and achieving a timetable for teachers with a minimum number of gaps are also crucial constraints for both private institutions and high schools. Therefore, this section is devoted to the summaries of previous work on high school timetabling.

In the early studies on the automation of high school timetabling, it is observed that lectures are sorted from the most constrained lecture, the most difficult lecture to place on the timetable, to the least constrained one. Then those lectures are placed on the timetable succesively beginning from the most constrained lecture up to the least constrained one by applying certain heuristics. In the work of Schaerf (1996), all such techniques are called direct heuristics and sample studies are illustrated (Schmidt 1979, Junginger 1986). Successive augmentation, the method of extending a partial timetable until all lectures are placed on it, has also been improved by the addition of local search and backtracking (Müller 2002).

Many techniques such as constraint programming, simulated annealing, tabu search, genetic algorithms and hybrid approaches have been applied to the problem of high school timetabling.

3.1 Constraint-Programming

Marte (2003) described and experimented with several constraint-based solvers in his work to solve the timetabling problem at German secondary schools of the gymnasium type. As stated in the work of Marte (2000), the details about constraint programming can be found in the study of Hentenryck et. al. (1997).

As Marte (2003) summarizes, a study that combines constraint propagation with a greedy algorithm along with local repair was carried out for the timetabling problem at Japanese high schools and was reported in [Yoshikawa 96].

In his statement of high school timetabling problem, the number of classes is low and each class has its own room. Hence, there is no need for the determination of room assignment except for lessons that require physical education equipment, science labs, etc. It is also assumed that all the lessons are taught in one building, which causes travelling times to be neglected.

Marte (2003) mentions that gymnasiums resemble high schools in the lower grades and universities in the higher grades although compactness should still be achieved.

In the study of Marte (2003), the fundamental aim is to transform the high-level timetabling problem into constraint models in terms of finite constraint networks, especially by the use of global constraints. As input data, a problem generator was developed and fed with the detailed school descriptions of ten schools, which caused a sample of 1000 problems to be generated for each school.

Constraint programming requires defining constraints over variables to solve a combinatorial problem. These constraints restrict the values that subsets of variables can take simultaneously. After achieving a constraint model with constraints and variables to describe a problem, a search strategy is employed. Assigning a depth-first manner during the search leads to chronological-backtracking, where search nodes represent partial assignments to decision variables. In the application for the timetabling problem, Marte (2003) describes that at each such node, a task is chosen and scheduled and rooms are allocated.

Constraint propagation is also combined with this search technique to reduce the computational cost of the search. It takes place after each commitment that is issued to the constraint solver and is performed in a fixed-point manner (Marte 2003).

In the work of Marte (2003), a track parallelization problem was examined and two reductions for this problem inference in school timetabling was proposed. This helped to modify the existing model generator and to produce enhanced constraint models. The model generator was combined with a suitable timetabling engine to form a problem solver giving statistically reliable and practically relevant results.

3.2 Local Search

Local search techniques basically aim to improve the current problem instance iteratively until a stopping condition is met or a satisfying result is reached. However, they don't guarantee to find the optimal solution for the problem at hand.

3.2.1 Simulated Annealing

Simulated annealing is a local search technique that simulates the cooling of a collection of hot vibrating atoms. In simulated annealing, a random solution is created initially. Then the algorithm enters a loop to navigate

the search space. This navigation is guided by a parameter called temperature. At the beginning, temperature is set to a high value and it decreases in each iteration, which is called the cooling scheme. At each step of the algorithm, a move that will modify the current solution is generated and it is executed according to the following rule: If the move improves the current solution, it is accepted. Otherwise, it is accepted according to the time decreasing probability, i.e the temperature. When the temperature decreases to a value very close to 0, a move that worsens the current solution has approximately no chance to be executed. The system then enters a so-called frozen state and the solution at that state is a local optimum solution. Abramson (1991a) applied simulated annealing to school timetabling problem. In the study of Schaerf (1996), it is reported that experiments were carried out with simulated annealing and tabu search for the high-school timetabling problem and the results of those experiments can be found in the study of Schaerf et. al. (1995). Those results showed that the performance of tabu search for high-school timetabling had a quite clear dominance over that of simulated annealing.

3.2.2 Tabu Search Techniques

Tabu search algorithm starts with an initial solution on the search space and it enters a loop to navigate the search space. At each step of the loop, it explores a portion of the neighbourhood of the current solution. The exploration of a neighbour solution is performed via a move, i.e. a modification that transforms the current solution to the neighbour solution. The examined neighbour with the best value of the objective function is assigned as the current solution.

A tabu list, which contains moves that are forbidden to make, is maintained at each step to prevent the algorithm from cycling. This list is implemented as a fixed-size queue where a predefined number of the last accepted moves are kept in reverse order. When a new accepted move is

added to the queue, the oldest move of the queue is discarded and it can again be used for further exploration of the algorithm.

The maintenance of a tabu list may prevent good moves from being made. Therefore, a movement is allowed to loose its tabu status if improves the best solution found so far. This option is enabled by the application of an aspiration function. In addition, despite the use of a tabu list, the search process may become trapped in certain regions of the search space. Schaerf (1996) uses adaptive relaxation, where costs involved in the objective function are dinamically altered to navigate the search process to unexplored regions of the search space. Some other extensions for tabu search can be found in the work of Schaerf et. al. (2001).

In the study of Schaerf (1996), the results of application of tabu search algorithm to high-schools were compared with a hybrid algorithm employing both tabu search and randomized non-ascendent method(RNA). In RNA, a random neighbour solution is chosen at each step of the search algorithm, if it is better or equal to the current solution. The experiments showed that the use of RNA greatly improves the performance of the algorithm, even more significantly for larger schools. The authors report that their algorithm was able to find a feasible solution in a reasonable amount of time in all practical cases. The details of this hybrid algorithm can be found in the next subsection.

Santos et. al. (2004) applied a new tabu search heuristic with memory based diversification strategies to the timetabling problem in Brazilian high schools. They experimented with the set of instances originated from the work of Souza et. al. (2003).

They used a requirements matrix, where each element r_{ij} of the matrix indicates the number of lessons that teacher i shall teach for a class j. In their instances, compactness of the timetable is mandatory for all classes, whereas it is a desired feature for teachers. As another hard

constraint, teachers should be assigned lectures only on their available periods. Totally, Santos *et.al.* (2004) define 4 hard constraints and 3 soft constraints. Their objective function is calculated by the aid of weights that reflect the relative importance of their constraints.

Before the tabu search, they use a greedy randomized constructive procedure to create the initial solution to commence tabu search. They apply the principle of placing the most urgent lessons to the most appropriate periods during the constructive process (Santos et. al. 2004). After the construction phase, tabu search with a short-term memory, i.e. the tabu list, an aspiration criteria and a long-term memory is employed. The long-term memory contains the frequency of moves involving a given teacher and class. Each move, which is a swap of two values in the timetable of a teacher, resembles the atomic moves explained in the work of Schaerf (1996). The frequency of these moves are stored to be used in the diversification strategy for stimulating the execution of few explored moves. As a second diversification strategy, they also consider the teacher workload to bias moves involving teachers whose timetable changes can produce bigger modifications in the current solution.

The authors report three sets of experiments carried out in the work of Santos et. al. (2004). The implementation in the first set of experiments lacks a diversification strategy. The implementation in the second set of experiments only takes into account the frequency ratio of transitions, whereas the implementation in the third set of experiments also considers the workload of teachers for the diversification strategies. The second and third set of experiments outperformed the first set of experiments and the previous results on the same data, which were reported in the work of Souza et. al. (2003). Hence, the addition of informed diversification strategies to tabu search brought about a simple design, while it produced better results and performed faster than the hybrid algorithm proposed in the study of Souza et. al. (2003).

3.2.2.1 Hybrid Local Search Employing Hill Climbing and Tabu Search

Schaerf (1999) applied local search techniques such as hill climbing and tabu search to several versions of the educational timetabling problems. Among all, a technique employing hill climbing and tabu search one after another in a cycle and executing this tandem search in a specified number of cycles performed best over real world high-school data.

Hill climbing approach suffers from the possibility of getting stuck in a local optimum point in the search space since the cost function of an instance is always improved or left unchanged after each iteration. The notion of worsening the moves is introduced in simulated annealing and tabu search techniques, each of which are explained with possible extensions in the study of Schaerf et. al. (2001).

Schaerf et.al. (2001) examine school, course and examination timetabling problems. Among these problems, the school timetabling problem has additional constraints such as combining lectures of two or more classes, two or more teachers participating in a lecture and compactness of class schedule. Compactness of a class schedule requires that the timetable created for that class must not involve gaps between lectures. This is an extremely hard constraint to satisfy and is one of the aspects that make school timetabling different from university course or exam timetabling.

Schaerf et.al. report the results of the work of Schaerf (1999), where they experimented with local search techniques for the high school timetabling problem, in their work (Schaerf et. al. 2001). They assume that the timetable is an integer-valued matrix, in which rows represent different teachers and columns represent the weekly periods. Each entry contains the index of the class the teacher is teaching at that period.

Schaerf uses a mutation like swapping operator called an 'atomic move', which swaps the classes assigned to two different periods of a teacher (Schaerf 1996). Also, a 'double move', which is the application of two atomic moves sequentially, is defined on the directly represented timetable instance. In this operator, the second move tries to resolve the infeasibility created by the application of the first move.

The hybrid search strategy, which applies hill climbing and tabu search one after another and which is used in the work of Schaerf (1999), is explained in the work of Schaerf et.al. (2001). In this strategy, firstly, random initialization of the timetable occurs, while considering the requirement matrix. Then, hill climbing search uses double moves on the timetable instance during a predefined number of iterations until no further improvement is achieved. After hill climbing search is over, tabu search is carried out by using atomic moves. Once it makes a given number of moves without improving, it stops and the hybrid search algorithm continues with hill climbing. This cycle involving both search techniques can be repeated until a stopping criteria is met.

This hybrid search starts with hill climbing instead of tabu search since hill climbing produces good solutions within a shorter time. Hill climbing is also used after tabu search since it modifies the solution in a way even if it cannot improve it. Therefore, tabu search achieves a different direction to improve the solution. Schaerf et. al. state that their application of local search techniques worked well with high school timetabling problems, the details of which can be found in the study of Schaerf (1999).

3.3 Evolutionary Approach

3.3.1 Genetic Algorithm with Local Search

Colorni et. al. (1990) used genetic algorithms with local search for the timetabling problem at a Italian high-school. They represented the

individuals of the genetic algorithm as matrices. They applied their versions of generalized genetic operators on those matrices. They compared their algorithm with various versions of tabu search and simulated annealing and concluded that genetic algorithm with local search outperformed simulated annealing and genetic algorithm without local search, whereas their results on the given data were competing with those of tabu search with relaxation.

An individual in their GA was represented as a matrix where each row corresponds to a teacher and each column corresponds to an hour. Each element of this matrix was defined as a gene, whose allele could take values among the set of jobs specific to the teacher. They defined the infeasibilities of a timetable and used a filtering algorithm to totally or partially remove the infeasibilities caused by the application of their genetic operators.

In the work of Colorni (1990), a hierarchical structure for the objective function is achieved by dividing the constraints into three levels, i.e feasibility conditions, management conditions, single teachers conditions. User defined weights were assigned to the three levels of constraint violations according to their relevance in the objective function. During calculation, individuals with infeasibilities were given high penalties to use selective pressure to reduce the number of individuals with infeasibilities.

Their GA performs a fitness-based selection for mating. They defined a crossover operator suitable for their matrix representation. During crossover, they calculate local fitness functions, each of which evaluates the fitness of a single row of the timetables contained in the parents. Then, they take the fitter rows of the fitter parent and fill the remaining timetable of the new individual with rows from the latter parent. The second son is created using the remaining unused rows from both parents.

Their first mutation operator, called mutation of order k, swaps two sets of k contiguous genes found in a single row and performs this operation for each row of the timetable. The second mutation operator, day mutation, swaps the two days belonging to a teacher in the timetable.

Finally, they introduce a local search operator that swaps hours and days in the timetable to move it to a local optimum point in the search space.

They conducted experiments with different probabilities for mutation and crossover operators, with and without local search and with high or low penalty values for infeasibilities. They reported that GA with local search and with low infeasibility penalties is definitely superior to the other versions tested. Low infeasibility penalties are preferred since they allow more infeasible timetables to exist in the population and broaden the search region by maintaining more diversity among individuals.

3.3.2 Hybrid Evolutionary Algorithm With a Timetable Builder and Local Search

Bufe et. al. (2001) used a hybrid evolutionary approach that also employs local search techniques in the form of specific mutation operators to solve the timetabling problem at a German high school.

In their definition of the high school timetabling problem, an event (i.e. a course meeting) has a number of weekly hours and must be assigned a room and a time period for each of these required hours.

Two seperate events may be participated by the pupils of the same class and therefore should be scheduled within the same time periods. So, event groupings are introduced to allow such occurings. Also, more than one teacher may participate in an event. When constraints are defined, these two cases are taken into consideration.

In the study of Bufe et. al. (2001), Bufe et. al classify constraints as hard and soft constraints. Hard constraints require that each class, teacher and room must be assigned to at most one event per time slot. The unavailabilities of teachers, classes and rooms for certain periods are kept and dealt with as hard constraints. Infact, these hard constraints are what we generally come across as exclusion or preset constraints. There are also 6 soft constraints defined.

Bufe et. al. state that a feasible timetable must meet all the hard constraints. In addition, they point out that using direct representation for the individuals of the evolutionary algorithm leads to infeasible timetables. And the application of a genetic repair function on those timetable instances lowers the correlation between the parent timetables and the offsprings. Hence, Bufe et. al. employ a timetable builder to create a feasible timetable from the permutations of events stored in an individual. The individual contains the permutations of the events in its first half and the created timetable in its second half. The deterministic placing algorithm within the timetable builder also attempts to satisfy the soft constraints and may produce valid but partial timetables.

In the hybrid approach, some mutation operators act on the high-level representation of the timetables in the timetable builder, whereas some others are applied to the individuals in the population of event permutations.

The evolutionary algorithm has a swap mutation and a partial matching crossover. The high-level mutation operators in the timetabler builder are applied after the timetable is created from an individual and before it is placed in the second half of the individual. Those operators don't harm the feasibility of the timetable produced but may violate soft constraints. They aim to place the unplaced events after unplacing, placing or replacing an event or part of an event in the timetable. The authors claim that those operators perform an extensive parallel hill climbing search on the better individuals.

The evolutionary algorithm applies uniform parental selection and replaces the worst 40 percent of the population in each generation.

The fitness function tries to minimize the number of unplaced events, the number of violations for the soft constraints and the standard deviation concerning the soft constraints. Only soft constraints are considered in this function since the timetable builder algorithm produces only feasible timetables that meet all the hard constraints and the high-level mutation operators applied thereafter may only violate soft constraints.

In their experiments (Bufe et. al. 2001), Bufe et. al. created 4000 generations each time and kept a population size of 20. They had to suffer from an 12-hour long computation time due to their expensive fitness function. They carried out three types of experiments. In the first type of experiments, they only used the genotype operations of the evolutionary algorithm. In the second type, they employed the genotype only operations in the first 1200 generations and used only the phenotype mutations of the timetable builder in the next generations. In the last type, they only experimented with phenotypic mutation operators. The second type of experiments reached the best results. In these results, best fitness values, zero number of unplaced events, minimum number of gaps in class' and teachers' timetables were achieved. Bufe et. al. commented that the initial usage of genotype operations in the experiments results in better starting points stored in the individuals for the timetable builder, whereas the application of phenotype mutations helps to place almost all events and fills the gaps in the timetables. Hence, the hybrid approach used in the second type of experiments appears to be more useful, while still not good enough to be used in daily school practice as stated by the authors of Bufe et. al. (2001).

3.3.3 Constructive Evolutionary Approach

Filho et. al. (2001) attempted the timetabling problem for public schools in Brazil by a constructive genetic algorithm. As stated in the work of Filho et. al. (2001), the details of constructive genetic algorithm can be found in the study of Furtado et. al (1998).

In their list of constraints, compactness of room usage is stated as necessary. Therefore, they require that all rooms are occupied at any time slot. In addition, teachers should have minimum number of gaps in their timetables, which is identified as a soft constraint in their study. As other soft constraints, they define preset constraints for the teacher timetables. They assign and experiment with different weight values for those soft constraints. Furthermore, they divide the teachers into three priority levels according to their number of classes and overall time dedicated to school. Teachers in a higher level has a more chance of having their constraints satisfied.

Filho et. al. (2001) state the high school timetabling problem as a clustering problem to apply constructive genetic algorithm. They aim to map valid teacher-class pairs to each of the time slots. They represent those pairs as binary columns. In addition, they use schemata to represent individuals of the genetic algorithm, where the length of these strings is equal to the number of possible teacher-class pairs. Their genetic operators and evaluation functions work directly on the schemata represetation of individuals. The initial population contains 100 schemata that are generated randomly.

Filho et. al. (2001) claim that their results were promising for the real world high school data they experimented with. This data contained 4 problems: morning, afternoon, evening timetables for Gabriel school and a morning timetable for Massaro high school. There were 5x5, 5x5, 5x4, 5x4 timeslots and 220,377,386,122 preference constraints for the problems respectively. The satisfaction of total preferences was

approximately between %80 and %90 in the resulting timetables. The number of windows, i.e gaps in teacher timetables, was very low for the teachers in the first level as well.

CHAPTER 4

INTRODUCTION TO EVOLUTIONARY ALGORITHMS

Evolutionary algorithms (EAs) are inspired from the biological observations beginning from Charles Darwin's discoveries in the 19th century. According to Darwin's findings, biological organisms evolve as they breed new generations, which causes them to adapt to their environment. EAs try to mimic this procedure to solve optimization problems.

EAs search through a solution space by using a population of possible solutions and by evolving them to determine the optimal solution. In this way, an optimization problem can be transformed into a search of the best individuals within a population by encoding potential solutions of the problem on simple chromosome-like data structures(i.e individuals of the population). This encoding is called the representation.

4.1 Overview Of Genetic Algorithms

Genetic algorithms (GAs), which were introduced in the study of Holland (1975), are specific instances of evolutionary algorithms. The GA begins with the random creation of a set of possible solutions. Each individual in this population is commonly called a chromosome. Atomic elements that make up a chromosome are called genes. The value a gene can take is called the allele of that gene.

After initialization phase, each individual in the population is evaluated according to the quality of the solution it encodes. Then, better individuals in the population are stochastically chosen to be mated to

create new individuals with recombined genetic material. Those newly created individuals can further be mutated with a certain probability. As the size of the population increases in size, a process that mimics natural selection takes place so that fitter individuals have more chance to survive. This sequence of actions are iterated in a cycle.

The fitness of each individual in GA is explicitly determined by the application of a fitness function over the individual. Hence, how good a possible solution in the form of an individual is calculated. This step of calculations take place in the beginning of the evolutionary cycle mentioned above. Then, the reproduction operator, which allows fitter individuals to mate, is applied to the population. Therefore, the recombination of genes achieved by mating is performed among chromosomes that contribute most to the overall fitness. After the application of the reproduction operator, crossover and mutation operators, which create new individuals from old ones, are used. All the processes in GA aim to perform a search on the solution space by means of modifying the content of the population and to move the population to the areas of the search space where better solutions, i.e fitter individuals, can be found.

The generic GA with selection, crossover and mutation can be described in pseudocode as in Table 4.1. Below, the main components of GA are described in more detail.

4.2 Reproduction

After the mating of individuals occurs within an iteration of the GA, the population size is increased by the addition of newly produced offsprings. Generally, the GA tends to keep the population size constant not to increase its computational cost. Therefore, a reproduction scheme is applied to decide who will survive in the next generation among the current crowded population. Two common reproduction techniques used are generational reproduction and steady state reproduction.

4.2.1 Generational Reproduction

Generational reproduction allows merely offsprings to be alive in the next generation at each iteration. Therefore, the whole population is replaced by the offsprings. If GA produces two new offsprings after each mating of two individuals, this mating process should be carried on as much as half the population size. Hence, an offspring pool is created to replace with the old generation. Elitism, which is allowing some of the fittest individuals of each generation to survive, may be used with this reproduction scheme. This prevents the loss of the best individuals in the old generation by replacement.

4.2.2 Steady-State Reproduction

In steady-state reproduction, only the individuals with the worst performance are replaced. More specificly, the family created after mating of two individuals is added to the current generation and the worst two individuals are destroyed.

4.3 Selection

In GAs, determining which individuals will be selected for mating has resulted in several different selection techniques. Since the whole operation of GA is infact a search of the best individual(i.e. the best solution) in the space of possible individuals, it should both explore new areas of this search space and try to improve better individuals. This trade-off between exploration and exploitation of the search space must be dealt with by the selection method. Individual performance, its rank among the population or its spatial ordering may be respected for selection and decide the direction for the search. The following subsections summarize the main selection methods.

Table 4.1 The Basic Genetic Algorithm

```
//Initialize a population P of p chromosomes by choosing an allele for each
//gene in each chromosome
For i=1 to p do
        P_i = GenerateChromosome(i);
//Evaluate fitness value f_i of each chromosome i in P
For i=1 to p do
      f_i= FitnessFunction(P_i);
//Continue the evolutionary cycle until the stopping criterion is met
while (max {f_1,..,f_p} < FitnessThreshold) do
        //Probabilistically select s<p members of P to create a subpopulation
        //P' where fitter chromosomes have higher chance for being selected
        For i=1 to s do
              P'_i= SelectChromosome(P);
        //Probabilistically select (p-s)/2 pairs of chromosomes from P
        //where fitter chromosomes have higher chance for being selected
      For i=1 to (p-s)/2 do
      Begin
              c_1= SelectChromosome(P);
              c_2= SelectChromosome(P);
              //For each selected pair of chromosomes, generate two
              //offsprings by the application of a recombination operator, i.e
              //crossover
              //Add the new chromosomes to the subpopulation P'
                P'_{s+i} = ApplyCrossover(c_1,c_2).FirstOffspring;
              P'_{s+i+1} = ApplyCrossover(c_1,c_2).SecondOffspring;
      End
      //Probabilistically select a fraction m of p chromosomes in P'
      //for mutation
      For i=1 to m.p do
              ApplyMutation(SelectChromosome(P'));
      //Update the current population
      P = P'
      //Evaluate fitness value f_i of each chromosome i in P'
      For i=1 to p do
              f_i= FitnessFunction(P_i)
end while
```

4.3.1 Fitness-proportionate Selection

In fitness-proportionate selection, which also appears as fitness-based selection or roulette-wheel selection, each individual has a chance of mating directly proportional to its performance, i.e fitness. If we think of individuals of GA as slots in a roulette-wheel, this approach resembles selecting a slot by spinning this roulette-wheel, where the size of each slot is proportional with its fitness. The fitness values of individuals should be scaled so that the range of fitness values do not have a negative effect in the selection (Fang 1994).

4.3.2 Tournament Selection

The basic idea of tournament selection is to choose randomly a predefined number of individuals among the population and to select the best of them for mating with a high probability. Choosing merely among a portion of the population aims to decrease the computational cost of the selection procedure.

4.3.3 Rank selection

Instead of using the absolute fitness values for selection and coping with scaling them, the individuals of the population may be ranked according to their fitness values and their chance of mating becomes directly proportional with their ranking among the population. The study of Fang (1994) reports that Whitley (1989) introduced a bias term, which can be applied to increase the effect of ranking so that the fittest members have much more chance for mating then before. Also, in this way, the chance of less fit individuals for mating becomes lower.

4.4 Genetic Operators

The two traditional global search operators of a genetic algorithm are crossover and mutation operators, both of which are explained below.

4.4.1 Crossover

Crossover operator adjusts the genetic inheritance of the mating process. It decides how segments of parent chromosomes will be distributed over offspring chromosomes. During crossover, the genes of parent chromosomes are passed on to offsprings without distortion. The aim of crossover is both to preserve and combine the genes causing high fitness and create new individuals with different gene combinations to maintain diversity.

4.4.1.1 One-point Crossover

As the name implies, a point is chosen randomly over the parents' chromosomes. The first offspring inherits the first portion of the first parent's chromosome up to the seperation point and the second portion of the second parent's chromosome. The second offspring inherits the other uninherited portions of the parents' chromosomes. As the length of the chromosome increases, the diversity brought about by this operation decreases. Therefore, merely using such a crossover operator with a low mutation rate may not lead to an efficient search on the search space of possible individuals.

4.4.1.2 Two-point Crossover

In two-point crossover, two points are chosen this time over the parents' chromosomes. The first offspring inherits the first and last portion of the first parent's chromosome and the middle portion of the second parent's chromosome. The second offspring again inherits the uninherited portions of the parents' chromosomes. More generally, instead of two points, GA can choose up to a predefined number of points and follow a similar procedure to distribute parents' chromosome portions over the offsprings.

4.4.1.3 Uniform Crossover

The basic unit of inheritance in uniform crossover is a gene. Each gene of the first offspring is chosen among one of its parents' genes in the same place probabilistically. The second offspring inherits the genes not chosen for the first offspring. So, whether a gene of a parent will be transmitted to the first child will be determined by a probability p, where the gene of the latter parent has a probability of being passed (1-p). Uniform crossover is able to produce more diversed individuals, which improves the exploration capability of the GA. However, it may also disrupt gene segments causing high fitness, which may become an obstacle for further exploiting areas of the search space with fitter solutions. This disadvantage can be overcome by employing smart operators for local search of such areas.

4.4.2 Mutation

Generally, mutation operator chooses a gene and assign it a new allele, i.e a new value for that gene. In this way, individuals can obtain the alleles that do not appear in the initial population or alleles that were lost during selection. Mutation is often applied after crossover with a low probability. Assigning probabilities to the application of genetic operators enables non-deterministic search through the solution space. With the application of mutation operator, no subspace of the search space can have zero probability of being investigated. However, as stated in the study of Miranda et. al. (1999), a high probability of mutation(i.e a mutation probability of 0.5) can be harmful since it always leads to random search independently of crossover operator.

4.5 Fitness Function

The fitness function generally arises from the objective function of the optimization problem. For instance, in a timetabling problem, fitness function can be calculated by summing the frequency of violated

constraints. Then, the fitness function descreases with the desirability of the solutions.

4.6 Parameters

Population size, probabilities assigned to genetic operators, number of individuals selected, etc. are all crucial parameters whose settings determine the performance of the GA to a great extent. It is commonly accepted that a low population number, high crossover and low mutation probabilities offer better solutions.

4.7 The Schemata Theorem

A schema is a template to describe similar subsets of genes at certain chromosome positions. From the schemata theorem, we can infer that a particular schema is replicated in the next generation with respect to its relative average fitness function value in the population. So, new generations will have more copies of fitter schemata, whereas they will be comprised of less copies of schemata with average fitness below the population average. The survival of a schemata under recombination and mutation operators is also dependent upon whether their average fitness values are below or above population average. The schemata theorem arises from the building block hypothesis, which states that highly fit (and with short defining length) schemata form partial solutions to a problem and a GA will combine these building blocks leading to a better performance and to the optimum of the problem (Miranda et. al. 1999).

CHAPTER 5

LITERATURE REVIEW ON MEMETIC ALGORITHMS

5.1 Overview Of Memetic Algorithms

The term, memetic algorithm, was first used by Moscato et. al. (1992) to describe evolutionary algorithms that employ local improvement of individuals heavily for fine tuning the search, which is essential especially in complex combinatorial spaces. The basic idea that inspires memetic algorithms can be traced to the difference between a meme and a gene. Genes of an individual pass to the next generation according to the application of genetic operators such as reproduction, crossover and mutation without being processed or refined by the individual. However, the existance of memes bring about the adaptation of units of information by the individual during its life-time. This individual optimization or learning is achieved by intensive local refinement in memetic algorithms.

The concept of learning for an individual can be categorized into two main approaches. The first one is Lamarckian learning, where the genetic content of an individual can be modified by local optimization during its lifetime. In the second approach, local optimization is carried out merely to bias the search without changing the genetic material of individuals directly.

Memes, i.e local search operators, can be static, adaptive or self-generating. In the first case, the local search always performs the same operations, whereas in the latter case its parameters can be adapted. As a last option ,a meme can be generated from stratch.

The most basic MA can be pseudocoded as in the work of Krasnogor (2002b), which is shown in table 5.1.

Table 5.1 The Basic Memetic Algorithm

```
Memetic_Algorithm()
Begin
        t=0     /*Initialize generations*/
        Generate an initial population P(t)
        Repeat Until (Termination Criteria is met)
        Begin
                Recombine
                Mutate
                Improve by Local Search
                Select for Next Generation
                t=t+1
        End Repeat
        Return the Best Solution(s)
End
```

In memetic algorithms, a local optimizer can be applied to every offspring before it is inserted into the population. Or, local search can be performed before or after mutation as displayed in figure 5.1 taken from the study of Krasnogor (2003b).

In this way, if an offspring is outside the subspace of local optima, it can be repaired by the local search operator so that it lies at a local optimum (Radcliffe et. al. 1994). Figure 5.2 has been taken from the study of Radcliffe *et.al.* After recombination of parents X and Y, the child Z' lies outside the subspace of local optima. Therefore, a local optimizer is employed to repair the child and modify it so that it becomes Z, which lies at a local optima. This constitutes the basic idea of memetic algorithms.

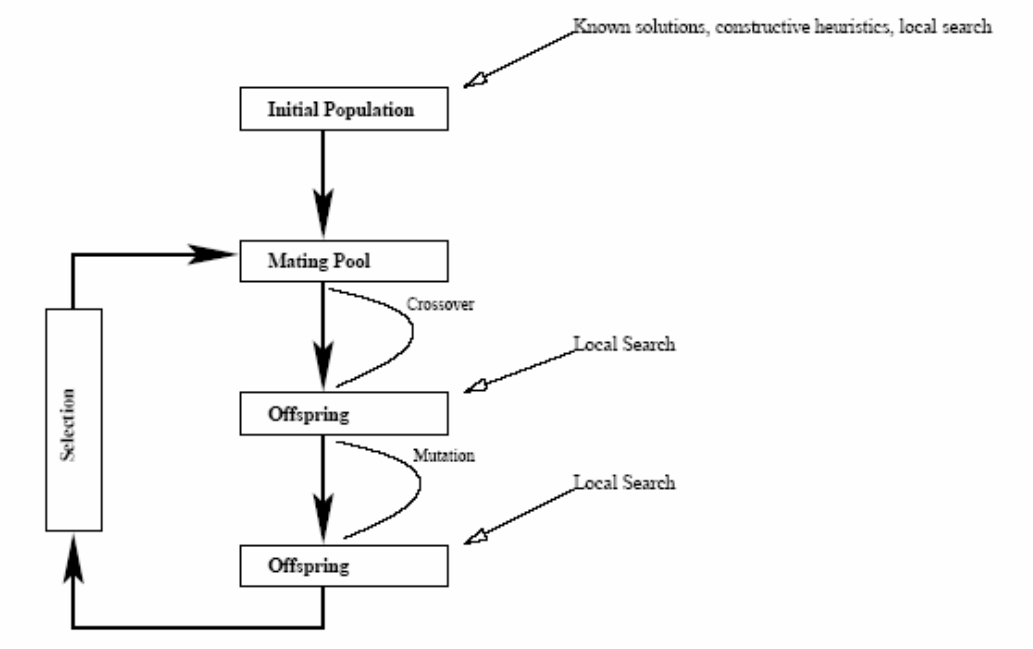

Figure 5.1 Diagramatic Representation of a Memetic Algorithm

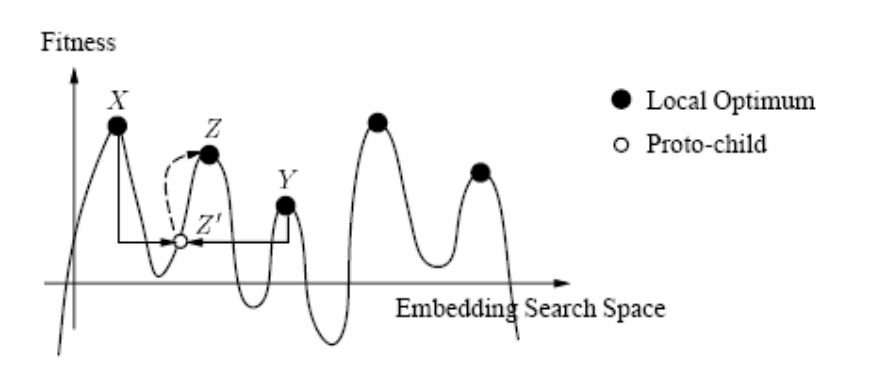

Figure 5.2 Memetic Algorithms' Search over the Subspace of Local Optima

Employing local search operators enable EAs to zoom-in good solutions by increasing their exploitation capabilities. However, MAs can suffer from premature convergence since local search may end up focusing a few good solution instances.

There are various strategies such as choosing which individuals to apply local search instead of applying it to whole population or using special operators to maintain diversity among the population.

The most crucial aspect to keep in mind when employing memes along with an evolutionary algorithm is that the act of memes and genes should be synergistic not detriemental.
In the work of Radcliffe et. al. (1994), a representation-independent form of a memetic algorithm along with N-point crossover, patching and hill climbing operators was introduced. Their simulation based on the application of the memetic algorithm to travelling salesman problem performed very well, whereas genetic algorithms were unsuccessful in solving the problem instances.

The main design issues for MAs, which are taken from the work of Krasnogor (2002a), can be listed as below:

1) When will we apply local search?
2) Which individuals will be applied local search? In the study of Krasnogor (2002b), it is pointed out that it could be a waste to apply local search operators to individuals far away from a global optimum.
3) How long will the local search be?
4) What acceptance criteria will the local search use?
5) How will the standard genetic operators be integrated with local search?
6) Shall we use a Balwinian or Lamarckian model for optimization?
7) How will we cope with multi-objective problems?
8) How can be avoid premature convergence?

The answers for these questions are discussed in the next section.

5.2 Design Of Memetic Algorithms For Timetabling Problems

In the work of Burke et. al. (2004), several frequently used strategies that can be applied to design memetic algorithms for timetabling and scheduling problems are discussed. Memetic algorithms speed up the process of exploitation with the aid of local search. The trade-off between this intensification and further exploration is achieved by maintaining a population in the algorithm.

Timetabling and scheduling problems are very difficult to solve because of several factors such as having huge search spaces, being highly constrained and difficult to represent and suffering from very-time consuming fitness evaluation. Therefore, the notion of self-improvement brought about by local search along with effective exploration owing to keeping a population of candidate solutions may help to cope with the difficulties of such problems.

As a first step in memetic algorithm design, Burke et.al discuss the decision of whether to allow infeasible solution instances or not. As stated in Erben's study (1995) they refer, genetic and local search operators that create only feasible instances or an eloborate representation such as representing groups of events in a gene can be employed if the first approach is chosen. However, they also mention that only working on the feasible regions of the search space may limit the explorative ability of the memetic algorithm.

As another way of dealing with infeasibility, repairing heuristics that are easy to implement and that do not reverse the changes made by other operators of the algorithm can be applied. As a last strategy to follow when infeasible solutions can be created by the genetic and local search operators, they mention heavily penalising infeasible solutions by a penalty allocating fitness function. Hence, such solutions have much less

chance to survive and thus to reproduce in the population. By referencing their previous studies (Burke et. al. 2001a, Burke et. al. 2001b), Burke *et.al.* claim that relatively low penalties for infeasibility should be assigned to prevent local search from recovering feasibility first and causing a potential increase in the violation of soft constraints. They also illustrate that penalty values can be adapted, i.e increased with the number of hard constraint violations.

In the work of Burke et. al. (2004), the authors discuss multi-phased strategies, delta evaluation for fitness functions and fitness landscape analysis for memetic algorithm design. They also emphasize that the application of genetic and local search operators should be balanced so that these two different sets of operators work in cooperation instead of working against each other. They figure out three main perspectives to adjust such a balance. These perspectives are listed as follows: the complexity of genetic and local search operators with respect to each other, the selection of solutions to apply each such group of operators and the execution time dedicated to each of those groups. For instance, their local search operators in their previous work, (Burke et. al. 2001b) were not powerful enough to improve newly created solutions since their recombination operators combined large parts from the parents. On the contrary, a local search that is too powerful can dominate the search and limit exploration. To overcome this, it can be decided to apply local search operators only on some better individuals or only after a number of generations.

Burke et.al also discuss that acceptance probability of solutions can be altered adaptively during the search. To illustrate, this probability can be dynamically adjusted as in the study of Burke et. al. (2001a) so that it decreases as the solutions in the population are more diversed and increases otherwise, i.e when the population converges. In this way, only better solutions are accepted when the spread of fitness is maintained whereas more non-improving solutions are accepted to cause diversification in case the population converges. Therefore, both

improvement on the fitness of individuals and prevention from premature convergence of population can be achieved.

By referencing the work of Alkan et. al. (2003), Burke *et. al.* state that different local search operators can be designed for each specific constraint or group of constraints. In addition, probability of applying a local search operator can be adapted dynamically according to the success of its previous progress. Moreover, a hyper-heuristic that determines the application of low-order heuristics can be assigned to decide which heuristic to use during the search.

Another idea for memetic algorithm design that Burke *et. al.* discuss is called the cooperative local seach which was proposed in the study of Landa Silva (2003). In this approach, each individual performs local search until it gets stuck and then recombines with some parts from the gene pool. Therefore, the notion of reproduction within generations is replaced by the self-improving life cycle of individuals, which also employs asynchronous cooperation, i.e recombination, between individuals.

Another study, (Burke et. al. 2003) discusses the usage of hyper-heuristics, which may be used to guide the low-order local search operators in a memetic algorithm.

5.2.1 Hyper-heuristics

In their work (2003), Burke et. al discuss the usage of hyper-heuristics as a way to handle a wider range of problem domains instead of customizing heuristics for a particular subset of problem instances. Hyper-heuristics operate at a higher level of abstraction with no knowledge of the domain and act on lower-level heuristics by deciding which heuristics to employ in a given situation. In this way, new problem

domains can be attempted by replacing the set of low level heuristics provided that an efficient hyper-heuristic algorithm has been developed.

The authors state that traditional meta-heuristics are either too simple to perform well or too knowledge-intensive to implement easily. However, it is the application of heuristics that bring about speed by narrowing the search space. The concept of hyper-heuristics arises from the idea of combining different heuristics in a way so that each of them compensates for the weaknesses of others.

Burke et. al references the work of Terashima et. al. (1999), which applies hyper-heuristic approach to large-scale university exam timetabling problems. In this work, they assumed a two-phase timetabling building algorithm. Genetic algorithms were employed to evolve the choices of heuristics used in the underlying algorithm and the condition that determines when to switch to a different phase of the timetable construction. The method was reported to solve even very large exam timetabling problems.

Another hyper-heuristic approach that Burke et. al discussed used an adaptive heuristic to improve on an initial heuristic ordering for exam timetabling problem. Initially, a solution was constructed by scheduling exams as determined by the original heuristic. If it was the case that an exam cannot be scheduled properly, it was scheduled by the order in a sunsequent construction. The process continued until all exams were acceptably scheduled or until a predefined time limit. The method could improve the quality when compared to the original heuristic.

Burke et.al conclude their work by commenting that employing the genetic algorithm for searching for a good algorithm might be better than employing it for searching a specific solution to a specific problem in timetabling.

5.3 Memetic Algorithms For Timetabling

Memetic algorithms have been successfully applied to various timetabling problems. Sections below summarize the most highlighting examples.

5.3.1 Mutation Operators of Varied Complexity and a Hill Climber for University Exam Timetabling

In the study of Burke et. al.(1995a), a memetic algorithm is used for university exam timetabling. In their study, they state that memetic algorithms can be more advantageous than genetic algorithms. As they illustrate, a possible solution instance, which was mutated to be brought in the scope of a local optima, can further be improved to reach local optima by the application of local search. They claim that the computational expense brought about by the execution of local search can be compensated by the reduction in the search areas that must be explored to reach a local optimum.

In their work, a timetable instance in the population is comprised of memes, each of which holds which exams are placed in each room in a period. In addition, there is a last meme that contains the unscheduled events since feasible but incomplete timetables can exist in the population.

They use the term meme instead of gene to refer to the unit of information on the chromosome. They state that memes are adapted by the individual whereas genes are passed unaltered. Since the units of a chromosome are altered by a hill climber after the application of genetic operators in their memetic algorithm, they do not use the term gene in their chromosome representation. They used heuristics along with random assignment to create an initial population that contains feasible instances with less penalties while maintaining sufficient diversity for the GA.

In their algorithm, light and heavy random mutations, i.e small and large scale alterations, are directly followed by a hill climbing operator. There is no recombination operator used. Instead, one of the two mutation operators is chosen to be applied on the individual. The light mutation operator picks a number of events from any period at random to reschedule them at other legal periods. The heavy mutation operator randomly reschedules all the exams in some periods of a timetable instance. This operator reveals well constructed periods and leaves them intact while considering each period of the instance in turn. Their deterministic hill climbing operator applies to each period of the timetable instance in a loop. At each step of the loop, the penalty of scheduling each event of the current period to every other period is calculated unless the scheduling causes any hard constraint violations. After that, each event is placed in the period causing least penalty. This operator also tries to schedule the unscheduled events.

Mutation and hill climbing operators are applied to create new individuals until the population expands to a specific size. Then, classic roulette-wheel selection is employed to choose individuals to generate a new population from the old expanded one. The fitness of each individual is directly proportional with the number of scheduled events and inversely proportional with the number of unscheduled events and conflicts between periods, which causes students to have exams one after another in the same day.

They experimented with real-life data taken from several universities. They observed that local search introduced to GA helps to find better solutions more quickly than random descent method. However, they also point out that the algorithm they proposed performs worse on more constrained problems.

5.3.2 Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation

In their study, Ross et. al (1994a) present a class of violation-directed mutation operators as main operators for memetic algorithms. In this work, they also describe the delta evaluation and how to measure the computational complexity of a timetabling EA in terms of what they call "evolutionary equivalents" instead of number of evaluations. In the study of Ross et. al. (1994a), they ignore constraints involving agents or places since they are easy to handle once the events/times timetable has been constructed in the exam timetabling case. However, they refer their work (Ross et. al. 1994b) for lecture timetabling, where such constraints become crucial.

Ross et. al. (1994a) use direct representation and a penalty-allocating fitness function. A penalty assigned for a constraint violation increases with the importance of the corresponding constraint. Ross et. al claim that delta evaluation employed during the fitness function calculations causes a significant speed up. In delta evaluation, the fitness function of a solution, i.e timetable t_h, is calculated by using an already-evaluated fitness belonging to one of its parents t_g (or a single parent if only mutation was used) and weighted sums of constraints involving a set of events whose assignments differ in t_g and t_h.

In their EA configuration, they always kept a population of size 1000. At each cycle of reproduction, they selected a parent and applied mutation on it. The new individual replaced the least fit one in the population provided that it was fitter than the worst among the population. With a probability of 0.2, they employed simple gene-wise mutation on the selected individual. This meant to consider each gene of the individual in turn and randomly reassign an allele to it with a very low probability (0.02). With probability 0.8, they applied their violation-directed mutation operator with a tournament size of 6.

Violation-directed mutation operators presented by Ross et. al. maintain violation scores for each event and its allele during fitness evaluation. This helps to reveal which events (alleles) in a timetable cause (would cause) more constraint violations and contribute (would contribute) more to the decrease in the overall fitness of the timetabling instance. Their directed mutation operators each choose a single event to mutate and an allele to mutate to. In case of events, this selection can either be choosing randomly among the worst events (i.e best candidates for mutation) or a probabilistic selection that is biased towards genes with higher violation scores. If the event is purely randomly selected, the directedness is expected to be achieved by allele selection. Allele selection can be made by using tournament selection among a set of k alleles to avoid the computational complexity brought about by other selection schemes.

The application of violation-directed mutation operators was held on a real-world examination timetabling data taken from a university. The problems involved assignments of events to available timeslots and the following constraints: exclusions, edge constraints and event-spread constraints. They also generated random problems both from random complete timetables and from real timetabling problems. To generate random problems from stratch, they initially construct a random complete timetable of 50 events scheduled within a predefined set of slots. Then, they generate edge and exclusion constraints that are satisfied in the timetable at hand. Finally, they construct the resulting problem by filtering some of the constraints generated.

In their study, Ross et.al (1994a) experimented with several variants of violation-directed mutation operators in a typical real-world exam timetabling problem and evaluated the performance of best such variants in randomly-generated problems as the number of constraints in the problem varied.

The results of experiments showed that choosing the allele in a stochastically biased way in directed mutation was obviously superior to applying other variants. This biased probabilistic allele selection turned out to give directed mutation its real power and effect, whereas the event selection need not be directed at all. In their experiments, the EA with directed mutation operators greatly overwhelmed the version with uniform crossover operator.

In their work, Ross et.al (1994a) describe the general examination/lecture timetabling problem and apply evolutionary algorithms on 5 real-world and 32 randomly generated timetabling data.

For the timetabling problem, they again use direct representation, where an assignment for each event is denoted by three consecutive genes on the chromosome. These three genes refer to time, place and agent assignments of the corresponding event. Their fitness function is inversely proportional with the weighted sum of occurences of constraint violations for each constraint type.

They suggest that constraints should be decomposed into lower order constraints (i.e constraints involving one or two events only) to prevent the fitness landscape from flattening. To illustrate, they point out that the constraint "no two exams that share common students should class" should be decomposed into seperate constraints as many as pairs of exams that share students. In addition, the penalty for each of those constraints must be arranged according to the number of students sharing the given exams.

Throughout their fitness evaluations, they use delta evaluation (Ross et. al. 1994a), which considers an already-evaluated similar timetable and changes in-between when calculating the fitness function of a new timetable. They employed a variant of violation-directed mutation operators examined in the study of Ross et. al. (1994a). In this operator, an event is randomly chosen and a new allele is assigned to it according

to the result of a tournament selection among 10 alleles for the event. The fitness of each allele in this tournament is assigned with respect to the degree to which they reduce the constraint violations on the event.

Their highly constrained real-world timetabling problems arise from examinations at two universities. Each of those problems had more than 400 edge-constraints on the average, while some of them also had a similar amount of event-spread or exclusion constraints. The EA configuration was the same as that explained in the work of Ross et. al. 1994a).

Ross et. al. (1994a) report the results of experiments with the particular EA they described. Those results reveal that their algorithm quickly finds perfect timetables for each of the real examination timetabling problems. They also emphasize the importance of their randomly-generated data as highly-constrained benchmark problems, especially for comparing techniques all of which have been successful in the real-world problems.

5.3.3 Violation Directed Hierarchical Hill Climbing For Timetabling

In [Alpay 03], memetic algorithms including violation directed mutations, crossovers and a violation directed hierarchical hill climbing(VDHC) method have been applied to the university course timetabling problem. Those new operators were introduced to prevent premature convergence that had occured in their real-world data. They considered only time assignments of course sections along with 7 types of constraints whose overall sum reaches up to approximately 3000. Each of those constraint types are one of the exclusion, preset constraints, edge-constraints or event-spread constraints.

They used direct representation, where each gene identifies a course section. In the chromosome, a hierarchical structure involving all course sections meetings grouped with respect to their courses, terms and finally departments is represented. Their fitness function allocates

weighted penalties multiplied by the occurences of constraint violations for each constraint in each constraint type. The initial population involves randomly created individuals all of which obey all the preset, exclusion and a specific type of event-spread constraints defined for course sections.

They named and experimented with new violation directed mutation operators in addition to the traditional mutation operator. Those directed mutation operators choose a term by rank selection, where the fitness of each term is based on the number of violations it contains. Then traditional mutation is applied on the term chosen. A variant of these operators uses rank selection to choose the allele for the gene to mutate, where the fitness of each allele is based on the number of violations it would cause in the term.

Since their timetable representation involved a hierarchy, they defined one-point and uniform crossover operators that treated certain blocks of genes (i.e a term) as a single gene in addition to the traditional crossover operators. Besides, they defined an operator that applied crossover only inside a single term. That single term was again chosen by rank selection and traditional crossover operators were applied to it once it was selected. Hence, they introduced several violation directed crossover operators that acted on different groupings of genes for their hierarchical representation.

The VDHC method proposed employs different operators for four of the constraint types. To apply the method on an individual, one of the four operators is chosen by a selection strategy biased towards those whose constraint type has more violations in the individual. Then the chosen operator attempts to resolve its corresponding violations with respect to the current resolution level of the VDHC method. The initial level of resolution requires that all the violations due to the related type of constraint be removed, whereas the second level considers violations in a selected block of genes and the third level merely attempts a single gene

in a block of genes. The selection scheme for those blocks is again biased towards ones causing more violations due to the constraint type of the operator. The resolution level is narrowed each time the operator fails to produce a better individual on the current resolution level. If the application of the VDHC method succeeds, it is reapplied on the individual.

In their EA configuration, they used elitism so that maximum fitness of the population can never reduce in next generations. On the real-world university timetable data, they experimented with different sets of their MA oparators by using both steady-state and trans-generational approaches. In this way, they aimed to figure out the best replacement strategy as well as the best set of MA operators. According to their results, the traditional crossover operator performed best among others, while the violation directed mutation operator that applied traditional mutation operator on a term rather than the individual overwhelmed its variants. Trans-generational approach for replacement produced much better results, where the average number of violations in the final generations of all runs was decreased to 0.2-0.3 with the best set of MA operators. Although the steady-state approach was rather successful with the VDHC, it couldn't find a solution in any of the runs without the VDHC operator no matter how many different strategies such as crowding, using weights during initialization were practised.

In the study of Ozcan et. al. (2005a), VDHC method was successfully applied to university final examination scheduling. In their application, there were three types of constraints defined. The first of these constraints required that no student should have conflicting examinations. The second type of constraints required that there must be a free slot between two examinations of a student when these two examinations are assigned on the same day. The last constraint imposed that maximum seat capacity must not be exceeded during a period. In the memetic algorithm proposed in the work of Ozcan et. al. (2005a),

random initialization of population, traditional one-point crossover, random mutation and swap operators are employed.

In the first stage experiments, Ozcan *et.al.* revealed that tournament mate selection, random mutation and trans-generational replacement strategy produce the best results when compared to their alternatives, which are ranking strategy, swap operator and steady-state replacement strategy respectively.

Having obtained the most successful configuration, the authors test the VDHC method on more data and report averaged success rates per run of approximately 100 percent for 10 different test cases.

CHAPTER 6

INTRODUCTION TO MULTIMEME ALGORITHMS

Memetic algorithms are evolutionary algorithms that employ an additional local search strategy as well as the common genetic operators such as crossover and mutation during the evolutionary cycle. Multimeme algorithms differ from memetic algorithms in that they self-adaptively choose which local search operator to use from the set of local searchers for different stages of the search or individuals in the population. Multimeme evolutionary algorithms were introduced in the work of Krasnogor et. al. (2001). In this study, they show how their multimeme algorithm is able to learn the best local search operator to apply to the individuals at different stages of the search. So, the optimum local search operator to apply is learnt during the evolutionary process. They also employ their algorithm to figure out which mutation operators to apply to individuals, where they regard their algorithm as an adaptive GA algorithm. In the work of Krasnogor et. al. (2001), they experiment with One-Max, NK-Landscapes and Travelling Salesman Problems(TSP). For the first two of these problems, their multimeme algorithm adaptively decides which mutation operators to use. For the last problem, optimum local search strategy is learnt by the algorithm.

In a multimeme algorithm, an individual comprises of both its genetic content and its memetic content. The genetic content represents a possible solution for the problem instance as in traditional evolutionary algorithms with direct representation. At any time during the evolutionary cycle of the multimeme algorithm, an individual also carries its own meme. The memes of an individual carry information that specifies which local improvement operators will be applied to the individual. They may

also specify the probability and/or the number of times for the application of such operators. All these composed memes in the individual make up the memeplex (Krasnogor 2002b). In the work of Krasnogor et. al. (2001), they also included the acceptance strategy to be employed in the local searcher in the meme for the multimeme algorithm solving the TSP problem. The acceptance strategy embedded in the meme of an individual can be one of the first-improvement or best-improvement acceptance strategies. In the best-improvement acceptance strategy, the local search operator is applied on the whole chromosome gene after gene, whereas the first-improvement strategy ceases once unsuccess occurs. The best-improvement strategy, however, takes backs its moves and continues with the next gene in case of unsuccess. In a multimeme algorithm, crossover and mutation operators are applied for genetic exchange and variety. Moreover, these operators also affect the memetic content of individuals as well as their genetic content. So, crossover operators of multimeme algorithms enable meme transmission, while their mutation operators can override the meme of an individual and assign a random local searcher among the set of available local search operators. During mutation, the frequency of this meme mutation is determined by a parameter, namely the innovation rate(IR) parameter. If this parameter is set to 0, a meme lost during meme transmission can never be re-introduced in the population. Therefore, IR value is set to a small value in the range [0,1] to guarantee a minimum level of exploration of the memetic space.

6.1 Meme Transmission

There are several ways of meme transmission in a population maintained by a multimeme algorithm as described in the work of Krasnogor (2002b). The first main way of meme transmission is called as vertical transmission or simple inheritance mechanism(SIM), where the offsprings inherit the memes of their fitter parent. algorithm for this process is given in the study of Krasnogor et. al. (2001) as in Table 6.1.

Table 6.1 Algorithm for SIM.

```
Individual_Level_Crossover(Parent p1, Parent p2, Offspring o1,
Offspring o2)
Begin
      CrossParentsGeneticMaterial(p1,p2,o1,o2)
      If  EqualMemes(p1->meme,p2->meme)) then
      begin
            CopyMeme(o1->meme,  p1->meme)
            CopyMeme(o2->meme,  p1->meme)
      End
      Else if p1->fitness==p2->fitness then
      Begin
            If FlipCoin() then
            Begin
                  CopyMeme(o1->meme,  p1->meme)
                  CopyMeme(o2->meme,  p1->meme)
            End
            Else
            Begin
                  CopyMeme(o1->meme,  p2->meme)
                  CopyMeme(o2->meme,  p2->meme)
            End
      End
      Else if (p1->fitness > p2->fitness) then
      Begin
            CopyMeme(o1->meme,  p1->meme)
            CopyMeme(o2->meme,  p1->meme)
      End
      Else
      Begin
            CopyMeme(o1->meme,  p2->meme)
            CopyMeme(o2->meme,  p2->meme)
      End
End
```

If the fitness of the parents are comparable and their memes differ, a random selection is made. In the second way, which is named longitudinal transmission, offsprings obtain memes from the individuals other than their parents. However, those memes can possibly bring about harm to their genetic content. The memes of an individual have an effect on its genes since they determine the application of local search operators on them.

6.2 Self-Generating Memetic Algorithms

In MAs, the local search operators can either pre-exist or be created and co-evolved by the MA (Krasnogor et. al 2003a). In the latter case, not only chromosomes, which represent possible solution instances as in any other EAs, but also local search operators, i.e memes, are evolved. In the work of Krasnogor et. al. (2003a), Krasnogor *et.al.* experiment with Maximum Contact Map Overlap Problem(MAX-CMO) to demonstrate the performance of self-generating memetic algorithms. In their study, they aim to provide local search with a new role: not merely a fine-tuner but a supplier of building-blocks.

The contact map they refer is an undirected graph that represents a protein's 3D fold, where each element that makes up the protein is a node and there is an edge between two nodes if they are neighbors. The problem they study, i.e MAX-CMO, tries to maximize the overlap number of two contact maps, which arises from an alignment between the two contact maps.

In order to figure out a metaheuristic that creates from scratch the appropriate local searcher to use under different circumtances, they explore the space of all possible memes by using a formal grammer that describes memeplexes. They also use genetic programming to evolve sentences generated from that grammer. In their study, they conclude that the success of their algorithm can be traced to a continuous supply

of building blocks and thus providing a more cooperative operation of local searchers and genetic operators.

CHAPTER 7

PROPOSED MEMETIC ALGORITHM FRAMEWORK FOR PRIVATE SCHOOL TIMETABLING

In this section, the details of the components for the memetic algorithm(MA) framework are explained. The main components of the MA that determine its success to solve an optimization problem are the representation of solution instances, initialization heuristics, the evaluation of violations for each type of constraints, a penalty-allocating fitness function, a replacement strategy, a mate selection method along with choices for other parameters and finally a set of useful local searchers in addition to the genetic operators. This last component mentioned is the main additional mechanism of memetic algorithms and it introduces a kind of intelligence to the search process. Basically, there can be only one local search method that is employed during a memetic algorithm. In this framework, there are 12 different local search operators, i.e hill climbers, that are applied to all the individuals of the population after mutation. In the proposed framework, local optimizers are controlled by a violation-directed hierarchical hill climbing operator(VDHC) as introduced in the studies of Alkan et.al. (2003) and Ozcan et. al. (2005a).

7.1 Representation

The application of evolutionary algorithms(EA), i.e genetic algorithms or memetic algorithms, to an optimization problem requires the appropriate choice of representation for the individuals in the population. In the below subsections, aspects that effect this choice in the proposed framework are discussed.

7.1.1 Direct vs. Indirect Representation

For the private school timetabling problem, this representation can be direct or indirect. When we use direct representation, each individual denotes a timetable. Hence, the genetic algorithm performs a search in the space of possible timetables directly. In the indirect representation, we can assign parameters to individuals to construct a timetable and then employ a timetable builder to create feasible timetables by using the individuals. In this framework, direct representation is used to avoid from the computational cost brought about by employing a timetable builder.

The usage of direct representation for any timetabling problem introduces two different choices. Each gene in the chromosome can refer to either a time slot or an event. If a gene refers to a timeslot and contains an event identification as an allele, we need genes as many as time slots in the chromosome. However, we then have the risk of losing or even duplicating certain events after the application of genetic operators. Hence, the individuals of the population tend to contain partial or invalid timetables, which causes a repair process to be employed. Fang (1994) addresses the problem of losing events in the timetables as label replacement problem and references (Abramson et. al. 1991b). In the proposed framework of this study, each course section meeting, i.e an event of the private school timetabling problem, is assigned to a place on the chromosome and has a value indicating its time slot mapping.

7.1.2 Determination of a Gene and Its Allele

In private school timetabling, the set of all course section meetings of the problem instance, i.e the set of all the events, is considered as a variable set. As stated before, private school timetabling problem is to assign feasible time mappings to each of those events while achieving a minimum number of constraint violations. As proposed in the study of Alkan et. al. (2003), the chromosome presents a hierararchical structure by preserving groupings within groupings. Course section meetings are at

the lowest level of this hierarchy. They are grouped with respect to their course sections, sections, divisions, grades and then branches (Figure 7.1). We can think each of those groupings as a classifier (Ozcan et. al. 2005a). With this hierarchical structure of the chromosome in mind, the content of a gene should be determined. A gene may be composed of all the course section meetings for a specific course section or merely a course section meeting. In this study, the first choice is applied to improve the performance of initialization heuristics as well as genetic operators.

The pair of each gene and its allele in the chromosome denotes the time slot assignments for meetings of a specific course section.

Either binary or integer values can be used to denote time mappings in the chromosome. This study chooses the latter approach since integer usage and operators defined accordingly have shown no deficiency when compared to those of binary values.

If we consider only time slots and neglect other resources such as rooms for any timetabling problem, the allele of each gene can be chosen among valid time slots. However, the need to assign available rooms to events as well as time slots causes a new dilemma. As a first method, each gene for an event mapping can contain a time-room tuple. In this way, supposing there are T time slots in a week and R rooms in a building, there are T x R possible mappings for an event. So, the allele of each gene is chosen among the set of integers beginning from 0 up to TxR-1. Hence, the computational cost is aimed to be reduced by dealing with only one integer instead of a tuple for an event. As a second method, we can alternatively divide the chromosome into two segments. In this second method, the first of the segments contains the time slot mappings for events, whereas the second half is comprised of room mappings for the events whose genes are allocated in the same order. Since room assignments do not cause significant problems in private school timetabling, they have been ignored in this study.

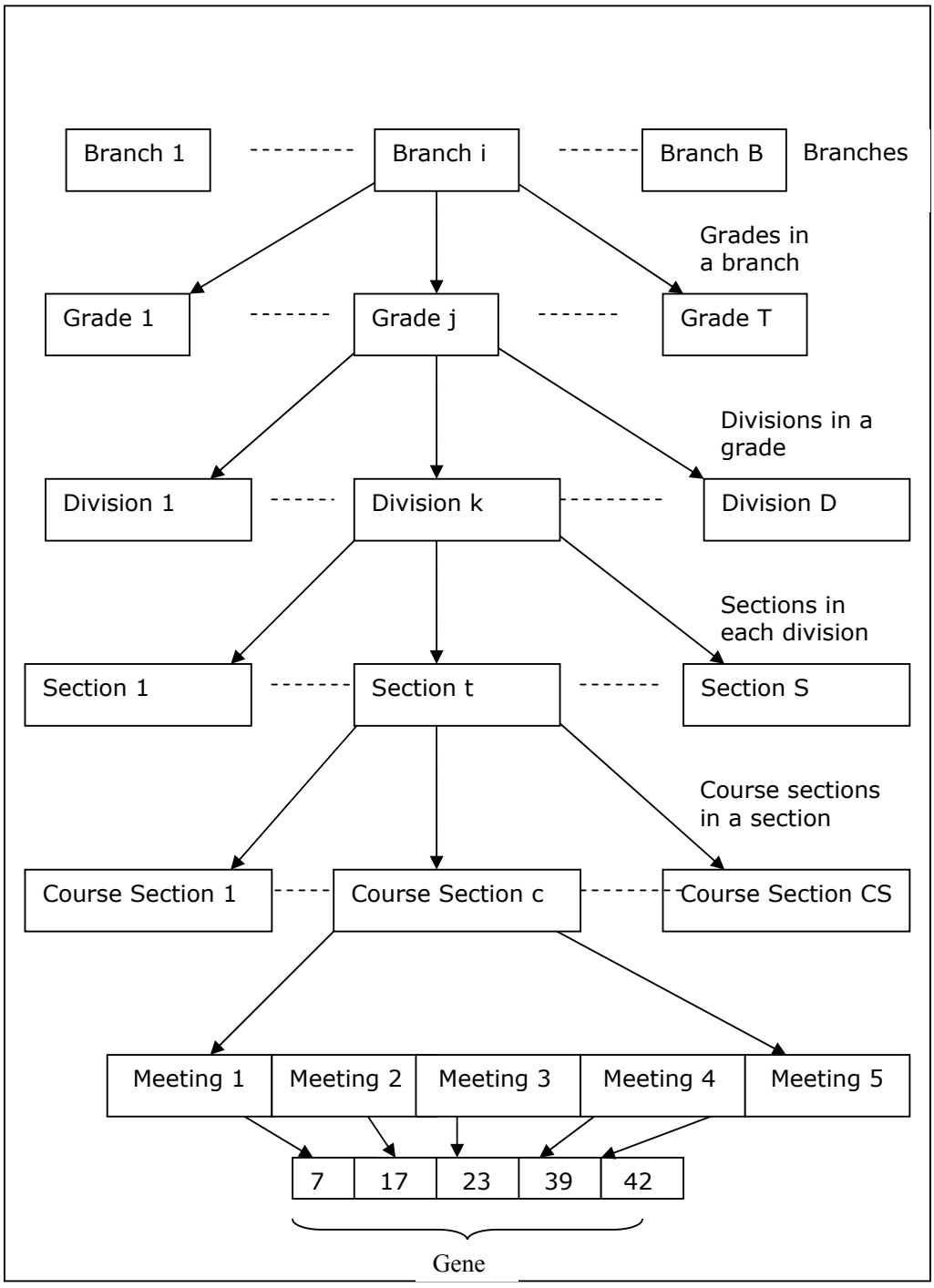

Figure 7.1 Individual Representation.

7.2 Initialization

At the beginning of the MA, a population that is comprised of possible timetable solutions for the problem instance is created. Individuals in the population are constructed by the initialization heuristics. According to those heuristics, individuals obey all kinds of preset and exclusion constraints(CS_PRE, S_EXC, S_PRE, I_EXC) as well as the first event-spread constraint(ES1), which requires that all meetings of a course section should be held at different days.

Firstly, a domain of available time slots for each course section is determined by examining its preset constraints, preset and exclusion constraints of its section, i.e its class, along with exclusions of its instructor. The allele assignment for a course section in the chromosome is then made by using its corresponding domain. Therefore, exclusion constraints for instructors and sections as well as preset constraints for sections and course sections are all met in the initial population. This is further explained in the next section. The aim of this approach is obviously to reduce the size of the search space, which becomes extremely large in the timetabling problems.

After the population is created, VDHC operator is applied over all the individuals. After that, the loop of producing next generations commences.

3. Constraints

As in other timetabling problems, there are two types of constraints in the private school timetabling. These are hard constraints, all of which must be satisfied for the timetable to be feasible, and soft constraints that denote preferences. Hard constraints for the private school timetabling can be determined as preset constraints, exclusions, edge constraints(EC1, EC2), the first event-spread constraint(ES1), workload constraints and most instructor constraints(I_MAXLOC, I_TRAVEL). The

remaining two constraints, namely minimum gap constraints for students and instructors, can be set as soft constraints and their violations may be permitted to some extent in the private school timetabling. In addition, the random data generator that is implemented and utilized for testing the proposed memetic framework produces problem instances that have at least one feasible solution satisfying all kinds of constraints except the minimum gap constraints. In other words, the solution produced by the random data generator normally does not have any hard constraint violations in it. However, in this solution, there may be gaps, i.e empty slots for the instructors or students, between the meetings assigned to them. Therefore, the MA implemented attempts to find a solution by giving equal importance to the satisfaction of all hard constraint types, whereas violation of minimum gap constraints may be permitted to some extent.

The definition and representation of constraints in the framework for timetabling greatly affects the computational burden put by the fitness evaluation. Among many others, there are two main methods for achieving the satisfaction of constraints. The first method assures that individuals of the population all satisfy certain constraints such as exclusions or preset constraints in each generation. This is guaranteed by initialization heuristics. Furthermore, the genetic operators and local searchers applied afterwards are designed so that they do not produce individuals that violate these constraints. For instance, common crossover operators such as one-point crossover or uniform crossover do not violate preset and exclusion constraints once they are satisfied in the initial generation. Besides, mutation operators are also designed so that they choose new alleles among available time slots for the gene to mutate. The second method leaves the computational burden of constraint evaluations to a penalty-allocating fitness function. This method is employed for edge constraints, most event-spread and instructor constraints. Both of these methods are applied as explained below.

7.3.1 Exclusions and Preset Constraints

There may be exclusion and/or preset constraints defined for instructors, sections or course sections. The basic idea to deal with such constraints is to minimize the search space as much as possible by the aid of initialization heuristics and appropriate implementation of genetic and local search operators.

The approach used is to construct a domain of mapping, i.e a domain that contains all the available time slots, for each course section while removing the excluded time slots from this domain. The domain of a course section is thus created by examining the excluded time slots of course section's instructor, the preset and excluded time slots of the course section's section and the course section itself. To illustrate, suppose there is a course PHYSICS-I offered to section M-1 by the instructor I11. Then, the pair (PHYSICS-I, M-1) denotes a specific course section, say CS_PHYSICS_M1. If the instructor I11 excludes time slots from s_i to s_j and the available time slots of section M-1 are the set of time slots from t_k to t_l, the domain of time slots for CS_PHYSICS_M1 becomes $\{t_k, ..., t_l\} - \{s_i, ..., s_j\}$.

Furthermore, the allele of the gene for each course section is made up of time slots from the course section’s domain of mapping throughout all the operations of the genetic algorithm such as crossover or mutation as well as initialization, which was also applied in the study of Fang (1994). In the work of Alkan et. al. (2003), preset constraints, exclusions and even a type of event spread constraints were satisfied during the initialization of the population.

This approach is supposed to work well if exclusion and preset constraints are defined as hard constraints that are not allowed to be violated in any case. However, if it’s not possible to create a timetable that satisfies all the given exclusion/preset constraints as well as other hard constraints and some of the exclusion/preset constraints appear as soft constraints,

we may leave their evaluation to the fitness function by the aid of weight assignments instead of the former direct approach. Nevertheless, removing burden on the fitness function evaluation by restricting the alleles for genes is considered to be a very efficient technique and is applied throughoutly in this framework.

7.3.2 Edge Constraints

Edge constraints in private school timetabling assure that each section (EC1) and each instructor (EC2) are assigned to at most one course-section at a given time period. These constraints are allowed to be violated on the individuals. If only those timetables that satisfy all the edge constraints are kept in the population, EA performance can be degraded. As stated in the work of Collingwood et. al. (1997), infeasible regions in the fitness landscape can help the search by providing inclined gradients towards good feasible solutions.

The violation of these constraints, whose number of occurings is calculated in the evaluation function, add up to the penalty assigned to the host individual. Therefore, the fitness of the individual is worsened, which makes it difficult for the individual to survive and thus to reproduce. Hence, natural selection simulated by the evolutionary algorithm causes the population to contain individuals with less number of violations for these constraint types.

Efficient methods to calculate the violations of these constraints are implemented. Generally, in a timetabling problem, a constraint matrix can be constructed to denote conflicting events, which are pairs of events that have an edge constraint defined on them. Then each event is paired with all the other events. If the paired events are conflicting and have the same allele for their time mappings, then the number of edge constraint violations is increased by one. The computational complexity of the overall check becomes proportional to N^2 if the total number of events is N. However, in the private school timetabling problem, we can design

another method whose computational complexity is proportional to N without the use of a constraint matrix.

To illustrate, the number of EC1 violations can be checked in one pass over the chromosome by counting the number of same time-slots used in the allele set of each section. All EC1 constraints belonging to a section can be satisfied if and only if every course section meeting of this section is assigned a unique time slot among the set of assigned time slots for this section.

The number of EC2 violations for an instructor can be found in a similar way by counting the instructor's course section meetings that are assigned at the same time slot.

7.3.3 Event-Spread Constraints

The first event-spread constraint (ES1), which requires that each meeting of a course section should be assigned to different days in a week, is satisfied during initialization of the population and after the application of genetic and local search operators.

Other event-spread constraints are checked as follows: While the edge constraint violations for a section are being calculated, a timetable is constructed for the section with its current time assignments. Within this timetable structure, all the constraint violations for the types mentioned above can be figured out.

7.3.4 Instructor Constraints

Violations for instructor constraints can be calculated by a method similar to the one for the other event-spread constraints. While the edge constraint violations for an instructor are being calculated, a timetable is constructed for the instructor with his/her current time assignments.

Within this timetable structure, all the constraint violations for the types mentioned above can be figured out.

7.4 Fitness Function

In memetic algorithms for timetabling problems, violation of constraints contribute to a decrease in the performance of the individual. Therefore, the performance of individuals can be calculated in terms of a penalty-allocating fitness function. This function is computed by assigning degrees of penalty in the form of weights for various types of constraint violations. Therefore, different types of constraint violations are penalized according to their relative importance. The fitness function then becomes

$$f(p) = 1 / (1 + \sum_{i=1}^{N} w_i \, c_i \, (p)) \qquad (7.1)$$

where p is an individual, N is the number of different constraint types, wi is the penalty assigned to the i^{th} constraint type and $c_i(p)$ is the number of violations for constraint type i on individual p. In the study of Fang (1994), it is stated that assigning higher penalty settings for a constraint type increases the artificial evolutionary pressure to remove such constraints from the population. Hence, higher penalty values can be assigned to hard constraints, which determine the feasibility of a solution, whereas relatively lower values may be assigned to soft constraints. Thus, penalty values of 1.0 are assigned for each hard constraint type and penalty values of 0.01 are assigned for minimum gap constraints in the proposed framework.

7.4.1 Fast Fitness Evaluation

The evaluation of individuals and the calculation of the fitness function is the bottleneck of evolutionary algorithms especially when timetabling problems are considered. In the timetabling problem, all the constraint violations that take place in the solution embedded in the individual must be figured out. In the work of Ross et. al. (1994a), delta evaluation,

which simplifies the computation for finding the number of constraint violations in the chromosome, is discussed. In their study, Ross et. al. (1994a) explain delta evaluation by considering two timetables g and h, which differ only in the assignments made to some subset D of the events E. If $P_C(t)$ is the weighted sum of violations belonging to C constraints for timetable t and C_D is the subset of constraints C which involve one or more events from the subset D of E, then P(h) can be calculated as:

$$P_C(h) = P_C(g) - P_{CD}(g) + P_{CD}(h) \qquad (7.2)$$

Therefore, the number of violations in timetable h can be expressed solely in terms of number of violations in timetable g and violations involving the D events. This idea is applied to the proposed framework as follows: The violations of constraints in the newly created individual after a hill climbing step can be calculated in terms of the violations in the initial individual before hill climbing, the initial number of violations in the portion of the chromosome to which the hill climber was applied and the new number of violations in that portion.

As it will be explained in the proceeding sections, the VDHC algorithm determines which hill climber will be applied to which portion of the individual during the local search phase. So, if we keep all the constraint violations that belong to each classifier located in the whole hierarchy, we can reduce the number of calculations for the fitness function. For instance, suppose the VDHC algorithm has chosen a specific instructor, say I_j, and the chosen hill climber will be applied to merely that instructors' genes on the current individual. After the application of the hill climber, only the contents of the individual for instructor I_j have changed. Let us denote the previous total number of constraint violations for instructors by $CV_i(I)$, where i refers to a constraint type among C constraint types, the new total number of constraint violations for instructors after hill climbing by $CV_i(I)'$, the previous number of violations for I_j by $CV_i(I_j)$, and the newly calculated number of violations for I_j by

$CV_i(I_j)'$. Then, the following calculation is sufficient to find out all the constraint violations of instructors in the chromosome:

$$\sum_{i=1}^{C} CV_i(I)' = CV_i(I) - CV_i(I_j)' + CV_i(I_j) \qquad (7.3)$$

As a result, owing to the VDHC operator that constrains the working area of hill climbers, we can only evaluate constraint violations in that area and calculate the whole number of violations by merely considering those violations since the remainings areas were left untouched during the local search. So, we needn't reevaluate the number of violations in those areas for the fitness evaluation after each local search step.

7.5 Genetic Operators

The two types of traditional genetic operators, namely crossover and mutation, are employed in the framework. The traditional crossover operators implemented are uniform crossover, one-point and two-point crossover operators. As explained before, they treat a course section with all its meetings as a single gene. The mutation operator applies the same approach. Moreover, while assigning a new allele to the randomly chosen gene, it chooses an allele that does not violate exclusion or preset constraints. In addition, this allele also obeys the first event-spread constraint(ES1).

In additional to the traditional genetic operators, new types of crossover and mutation operators have been implemented by making use of the hierarchical organization in the chromosome as explained in the study of Ozcan et. al. (2005a). These operators have been listed in Table 7.1 and Table 7.2.

The basic idea that authors of Ozcan et. al. (2005a) address is that a classifier at any level of an arrangement can be chosen as a single unit. For instance, all the course section meetings of an instructor or all the course section meetings of a section can be chosen as a single gene.

Hence, operators can be applied on such a classifier as if it were a single gene. In the first set of mutation operators implemented in the framework, a static subgrouping on the chromosome, i.e a classifier, is randomly chosen as the area of concern for the operator. To illustrate, branches, grades, divisions, sections or instructors are each used for these grouping of genes, i.e course section meetings or shortly events. Then, traditional mutation is applied within that classifier of length L with mutation rate equal to 1/L. The second set of mutation operators are similar to the first set except that they apply traditional mutation in each of their corresponding classifiers rather than choosing only one. The third set of mutation operators are violation-directed. In other words, the probability of choosing a classifier to operate on is proportional to the number of its constraint violations in it. Firstly, the subgroup that causes more violations is identified via a selection strategy, namely the tournament selection, then the genetic operator is applied on only the regions belonging to the subgrouping. For example, VD_INSTRUCTOR_MT chooses an instructor whose course section meetings cause more number of violations and applies traditional mutation with mutation probability 1/L on them, where L is the number of course sections that the chosen instructor teaches.

In the new one-point crossover operators defined as violation-directed operators, crossover points are chosen among the ones that cause more constraint violations. In both new uniform and one-point crossover operators listed, a grouping within the chromosome is regarded as a gene and operators are applied accordingly. Branches, grades, divisions or sections are used for these grouping of genes, i.e course section meetings or shortly events. For instance, if the basic unit for uniform crossover is chosen as a section, i.e a class, then offsprings are produced as follows: For each section, the whole section of a probabilistically chosen parent is copied to one of the offsprings and the corresponding section on the other parent is copied to the other offspring. In short, this process treats each section as a gene. A similar approach is also implemented for other possible groupings such as branches, grades, etc.

Table 7.1 New Mutation Operators in the Framework

Operator Name	Single Unit	Choice of a Unit	Applies to All Units
COURSESECTION_MT	Course Section	random	yes
RAND_SECTION_MT	Section	random	no
RAND_DIVISION_MT	Division	random	no
RAND_GRADE_MT	Grade	random	no
RAND_BRANCH_MT	Branch	random	no
RAND_INSTRUCTOR_MT	Instructor	random	no
ALL_SECTION_MT	Section	all units	yes
ALL_DIVISION_MT	Division	all units	yes
ALL_GRADE_MT	Grade	all units	yes
ALL_BRANCH_MT	Branch	all units	yes
ALL_INSTRUCTOR_MT	Instructor	all units	yes
VD_SECTION_MT	Section	violation-directed	no
VD_DIVISION_MT	Division	violation-directed	no
VD_GRADE_MT	Grade	violation-directed	no
VD_BRANCH_MT	Branch	violation-directed	no
VD_INSTRUCTOR_MT	Instructor	violation-directed	no

Table 7.2 New Crossover Operators in the Framework. The UX suffix in the operator name denotes that the operator is a uniform crossover and the 1PTX suffix shows the operator is a one-point crossover.

Operator Name	Single Unit	CP Choice
COURSESECTION_UX	Course section	
COURSESECTION_1PTX	Course section	-
SECTION_UX	Section	-
DIVISION_UX	Division	-
GRADE_UX	Grade	-
BRANCH_UX	Branch	-
I_UX	Instructor	-
VD_COURSESECTION_1PTX	Course Section	violation-directed
VD_SECTION_1PTX	Section	violation-directed
VD_DIVISION_1PTX	Division	violation-directed
VD_GRADE_1PTX	Grade	violation-directed
VD_BRANCH_1PTX	Branch	violation-directed
VD_I_1PTX	Instructor	violation-directed

Each of the above operator types mentioned above are implemented and their contribution to the overall memetic framework is investigated during the experiments.

7.6 Mate Selection and Replacement Strategies

In several studies (Alkan et. al. 2003, Ozcan et. al. 2005a), it is reported that tournament mate selection performed better than ranking strategy for timetabling problem instances. In addition, trans-generational replacement strategy, where the entire population except two best individuals is replaced with the offspring pool in each generation, outperfomed steady-state replacement, where only two individuals are selected and two offsprings are produced. Hence, these successful strategies, namely the tournament mate selection method and trans-generational replacement strategy, are used in this study.

7.7 Low-order Local Search Operators (Hill Climbers)

There are 12 different local search operators in the framework. Their execution is controlled by the VDHC operator, which chooses one of the hill climbers to apply on the current individual. The VDHC operator also decides which portion of the current chromosome will be given to the hill climber as its work area.

Each of the hill climbers attempts to resolve constraints of its type. Once a hill climber is invoked with a part of the individual's chromosome, it processes the course sections in the given chromosome part one by one. It chooses the next course section, i.e gene, to consider in a random fashion. Firstly, it counts the number of constraints belonging to its target constraint type. For instance, if the hill climber is HC_S_MINGAP, which tries to minimize the number of gaps between course section meetings of a section, i.e a class, it counts the number of gaps that are next to the meetings of the current course section. The aim here is to figure out the contribution of the current course section to the constraint violations of

the hill climber's constraint type. Then, the hill climber attempts to resolve its corresponding constraint violations caused by the current course section via intelligent reassignments of its meetings. After this process, the hill climber again counts the number of constraint violations that belong to its target constraints and that are caused by the current course section. If the course section now causes less violations of target constraint type for the hill climber, the moves of the hill climber on the current gene is accepted. Otherwise, the moves are taken back and the initial assignments for the current course section are preserved. Afterwards, a new course section is chosen randomly and the hill climbing moves continue to perform on the given chromosome portion. While considering the success of a hill climber's move, overall fitness evaluation is not performed since it would greatly increase the computational time.

The generic algorithm for the implemented hill climbers is given in table 7.3. The hill climbers all have different hill climbing moves. A naive approach is to randomly reassing the meetings of the current course section. However, the implemented hill climbers have far more intelligent moves than merely random assignments.

In its hill climbing move, HC_EC1 tries to assign a course section's meetings to empty slots on it's section's timetable if they clash with other meetings of the the section. This operator marks all the slots that are occupied by the other meetings of the current section and tries to assign the meetings of the current course section to unmarked available slots for the section. This hill climber, like all the other hill climber operators, obeys all kinds of preset and exclusion constraints as well as the first event-spread constraint(ES1) while performing reassignments for the current course section. As a hill climbing move, HC_EC2 assigns a course section's meetings to empty slots on it's instructor's timetable if they clash with the other meetings of the instructor.

Before the reassignments for a course section, HC_S_MAXWL finds the workload of the current course section's section for each of its days. It

then sorts the section's days in increasing order according to their workloads. Then, if a meeting of the current course section is on a day whose daily workload exceeds the section's maximum daily workload, that meeting is reassigned to an empty time slot on a day having less workload on the section's timetable. Therefore, heavy workload on days is relaxed by assigning the meetings on those days to other slots where the workload is less for the section. HC_I_MAXWL operates in the same way, whereas it considers the maximum workload of the current course section's instructor.

Table 7.3 Generic Algorithm of a Hill Climber in the Framework.

```
HillClimber(CurrentLevel)
//The Current Level indicates a specific classifier,
//a course section, a section, a division, a grade, a branch or an
//instructor

// A course section with its meetings comprise a gene.
Gene TempGene
Gene CurrentEvent

For i=0 to EventNo in current level do
      //Choose an event to consider
      CurrentEvent = ChooseEvent(CurrentLevel)

      //Count number of violations that violate hill climber's target
      //constraints and that are caused by the current event
      PreviousViolations = CountViolations(CurrentEvent);

      Copy CurrentEvent's Gene to TempGene

      //Attempt to resolve violations that violate hill climber's target
      //constraints and that are caused by the current event by heuristically
      //reassigning meetings
      HillClimbingMove(CurrentEvent)

      //Count number of violations of hill climber's target constraints
      //caused by the current event
      NewViolations = CountViolations(CurrentEvent);
      if NewViolations >= PreviousViolations then       //No Improvement
            Copy TempGene to Event i's Gene
End for
```

HC_S_MINWL also finds the daily workloads of the current course section's section. If a meeting of the current course section is on a day that has workload below the minimum daily workload defined for the section, another meeting that belongs to a different course section of the current section and that is on a day that has more daily workload is chosen. That meeting is assigned to an available slot on the day of the current course section's meeting. Therefore, this hill climber tries to increase workload on days that have daily workload below minimum. The operation of HC_S_MAXWL and HC_S_MINWL thus coorporate.

HC_I_MINWL is similar to HC_S_MINWL. However, it considers the minimum daily workload of the current course section's instructor.

HC_S_DIVMAXWL endeavors to resolve violations of divisional maximum daily workload constraints defined for the sections. To perform its hill climbing move, the hill climber finds the divisional daily workload values belonging to the division D that offers the current course section for the section. Once those values are found, the operation is similar to that of HC_S_MAXWL. In this case, a meeting of the current course section is reassigned to an empty time slot on a day having less divisional workload for D on the section's timetable if it is on a day whose total divisional workload for D exceeds the section's maximum divisional workload for D. For instance, suppose, in a day, a section can be assigned to at most 3 meetings of courses offered from the verbal division. If the total number of meetings of courses offered by the verbal division for this section exceeds 3 on a day, one of those meetings is reassigned to another day where there are less courses from the verbal division in one move of HC_S_DIVMAXWL.

HC_S_DIVMINWL also calculates the daily divisional workload values in the timetable of the section that is assigned to the current course section chosen in the hill climbing process. Again, if a meeting of this course section offered from division D is on a day, say day 0, where the workload belonging to its division D is low than the minimum value

allowed for D, a meeting of another course section that is assigned to a day where the workload caused by courses of D is larger is found. That meeting is assigned to day 0.

Other hill climbers that make use of heuristics in their moves are HC_S_MINGAP and HC_I_MINGAP, whose moves are similar to each other. In their hill climbing move, if a meeting of the current course section is next to a gap, i.e an empty slot for its section(or for its instructor in HC_I_MINGAP), it is reassigned so that the gap is removed. If the gap is below the meeting's slot, the meeting is assigned to an earlier slot to remove the gap. If the gap is above the slot assigned for the meeting, the meeting is assigned to a later slot, which will remove the gap. If there is gap both below and above the meeting considered, a random choice is made. According to this random choice, the meeting is assigned to either an earlier or a later slot and the gap is removed.

7.8 The VDHC Method

This method is applied to all the individuals of the population after initialization and after the application of genetic operators during each generation. This method has been successfully applied to various timetabling problems such as nurse rostering, final exam scheduling and university course timetabling in many studies (Alkan et. al. 2003, Ozcan et. al. 2005a, Ozcan et. al. 2005b). For the implementation of this method to solve the private school timetabling problem, a hill climbing operator, i.e a local searcher, is defined for each constraint type whose violations are summed in the fitness function. For instance, there are different hill climbers for EC1 and EC2 constraints. The VDHC operator, which can be comprehended as an upper-level local search operator, coordinates the process of lower-level hill climbers. It decides when a specific hill climbing operator will act on the chromosome of the individual that is being improved. It also determines the region of the chromosome that the hill climbing operator will act on. The chosen hill climber endeavors to resolve its type of constraints on each course section by

choosing them randomly. If it is unable to resolve violations for a course section, it takes its moves back for that course section and passes on to the next course section. Therefore, the hill climbers in the framework have been implemented somewhat in a random-move fashion. After the process of the current hill climber finishes, VDHC evaluates the modified individual. If its fitness is improved after the modification of the hill climber, VDHC accepts this modification and the individual has changed. Otherwise, VDHC preserves the initial individual. As a result, the application of VDHC operator to the individual cannot reduce its overall fitness value.

The part of the chromosome where the chosen hill climber will be applied depends on the current resolution level of the algorithm. The initial resolution level contains the whole chromosome. While the current hill climber performs successfully on the individual, the algorithm stays in the same level and endeavors to improve the whole individual. If the hill climber fails to improve the overall fitness of the individual after it has completed its work, the resolution level is lowered by one. Therefore, the current resolution level now denotes a specific branch of the private school. Thus, only that part of the chromosome is fed to the chosen hill climber. The highest resolution level indicates the whole chromosome and the lowest level denotes merely a course section. As a result, the chosen local optimizer initially attempts to improve the current portion of the chromosome. If it fails, it acts on a smaller portion and so on. A new hill climber operator can be employed after the process of the previous operator. In this case, a hilclimber that tries to resolve constraints that are higher in number on the current individual is chosen.

To analyze how a hill climber is choicen for the private school timetabling problem, we should further consider the organization of a private school timetabling instance. In the private school timetabling problem, the 12 constraint types whose violations are allocated in the fitness evaluation can be divided into two groups. These groups are constraints for sections and constraints for instructors. Similarly, the 12 hill climbers used in the

private school timetabling framework can be divided into two groups as well. The first group of hill climbers all attempt to resolve violations of constraints defined on sections. Those constraints are edge constraints, workload constraints and minimum number of gap constraints for sections. The second group of hill climbers try to resolve constraints defined for the instructors such as edge constraints, workload, minimum gap or minimum travelling times constraints. Therefore, the organization of the chromosome infact encloses two different types of hierarchy levels. The first of these hierarchies groups course sections into sections, divisions, grades and branches. The second one groups course sections into instructors. Hence, classifiers of group 0 are course sections, sections, divisions, grades and branches. Classifiers of group 1 are course sections and instructors. Once the organizational aspects are understood, we can analyze the function that chooses a hill climber(Table 7.4). The VDHC steps continue until a predefined number of iterations. Maximum number of unsuccessful iterations is kept as 10. Maximum number of successful iterations is kept as the chromosome length. The VDHC algorithm is summarized in tables 10.5 and 10.6.

Table 7.4 Algorithm for Choosing a Hill Climber in VDHC

```
HillClimber_ID ChooseHillClimber(p, temp_i, current_level)
    //The choose function below performs a tournament selection
    //with tour size 2.

    //It chooses a constraint type that is more often violated on the
    //inputted portion of the chromosome and returns its
    //corresponding hill climber.

    Begin
        if current_level is whole_chromosome then
                Choose a constraint type that is more often violated
                 on the chromosome and return its hill climber
        else if current_level is a classifier of group 0
        (i.e a branch is specified) then
        begin
                if there are violations of constraints belonging to
                group 0 in current level then
                        Choose a constraint type that is more often
                        violated on the specified classifier and return its
                        hill climber(a group 0 hill climber)
                else
                begin
                        initialize current level to whole chromosome
                        Choose a constraint type that is more often
                        violated on the chromosome and return its
                        hill climber
                end
        end
        else if current_level is a classifier of group 1
        (i.e an instructor is specified) then
        begin
                if there are violations of constraints belonging to
                group 1 in current level then
                begin
                        Choose a constraint type that is more often
                        violated on the specified classifier and
                        return its hill climber (a group 1 hill climber)
                end
                else
                begin
                        initialize current level to whole chromosome
                        Choose a constraint type that is more often
                        violated on the chromosome and return its
                        hill climber
                end
        end
    End
```

Table 7.5 The VDHC Method

```
VDHC(Individual i, Parameters p)
begin
    CopyIndividual(i, temp_i)    //Copy the individual to temp_i
    EvaluateIndividual(temp_i) //Evaluate the fitness of the individual
    previous_fitness = temp_i->fitness_value

    //The initial level of the algorithm is the highest level that
    //corresponds to the whole chromosome

    current_level = whole_chromosome
    current_successful_iterations = 0
    current_unsuccessful_iterations = 0
    max_successful_iterations = p->chromosome_length
    max_unsuccessful_iterations = 10
    while(current_successful_iterations <max_successful_iterations and
        current_unsuccessful_iterations <max_unsuccessful_iterations) do

    begin
      // Choose a HC Method that tries to resolve constraints that
      //have more violations on the current level of the individual
       CurrentHillClimbingMethod = ChooseHillClimber(p, temp_i,
                                                    current_level)
      //Current level denotes which portion of the chromosome will be
      //attempted for optimization

      temp_i = ApplyCurrentHillClimber(p, current_level,
                                       CurrentHillClimbingMethod,
                                       temp_i)
      EvaluateIndividual(temp_i)
      if( Better(temp_i->fitness, previous_fitness)) then
      begin
          CopyIndividual(temp_i, i)
          prev_fitness = temp_i->fitness_value
          current_successful_iterations++
      end
      else
      begin
          //Current application of hill climber may have corrupted the
          //individual, so take its moves back and acquire the content of
          //the individual that has been achieved after the last successful
          //modification
          CopyIndividual(i, temp_i);
          current_unsuccessful_iterations++;
          //Find a sublevel that causes more violations for the current
          //constraint type to be improved
          //In other words, restrict the area of concern

          current_level = LowerLevelByOne(CurrentHillClimbingMethod,
                                current_level);
      end
    end while
end VDHC
```

Table 7.6 Function That Restricts the Portion of Current Chromosome in VDHC.

```
LowerLevelByOne(HillClimberInfo CurrentHillClimbingMethod,
                    LevelInfo current_level)
begin
    if current_level is already the lowest level(i.e a course section) then
            Assign current_level to whole chromosome
    else if CurrentHillClimbingMethod is of group 0
    (i.e a HC for a constraint defined on a section) then
    begin
            Find a classifier in current level that has more violations of
            constraints belonging to group 0.
            Assign current_level to this classifier
    end
    else if CurrentHillClimbingMethod is of group 1
    (i.e a HC for a constraint defined on an instructor) then
    begin
            Find a classifier in current level that has more violations of
            constraints belonging to group 1.
            Assign current_level to this classifier
    end
end
```

CHAPTER 8

MULTIMEME ALGORITHM FOR PRIVATE SCHOOL TIMETABLING

Multimeme algorithms have also been implemented for the private school timetabling problem. In the proposed framework, there are two versions of multimeme algorithms applied. In the first implementation, each individual carries only one meme from the pool of available memes. This meme denotes the hill climber to apply to the individual and the number of maximum unsuccessful iterations the hill climber will be applied to the individual. Since there are 12 different hill climbers and the unsuccessful iteration limit is kept between 5 and 10 during the creation of a random meme, there are 72 different meme configurations in the meme pool.

In the second implementation, the memeplex of an individual contains as many memes as the number of different local search operators. During initialization, a local search operator is assigned randomly to each meme in the memeplex. Each meme again contains information about how many times its corresponding local search operator can be applied unsuccessfully. As in the VDHC implementation, the maximum number of successful iterations is kept as the chromosome length. The improvement strategy to use for the operator and the probability of applying the local search operator can also be embedded in meme information. However, the random improvement strategy and a probability of 1.0 for applying the local search operator have been chosen for the multimeme algorithms. The memeplexes of individuals are evolved during the evolutionary cycle with crossover and mutation operators. After mutation, each of the individuals go through a hill climbing process (Table 8.1). In this process, all the local searchers that are referred in the memeplex of the individual are applied to the individual a number of times.

Table 8.1 The Hill Climbing Process for the Multimeme Algorithm

```
MMA_HillClimb(Individual i, Parameters p)
begin
        CopyIndividual(i, temp_i)          //Copy the individual to temp_i
        EvaluateIndividual(temp_i)         //Evaluate the fitness of the
                                           //individual
        previous_fitness = temp_i->fitness_value
        for x=0 to p->NoOfMemesInIndividual do
        begin
             current_successful_iterations = 0
             current_unsuccessful_iterations = 0
             max_successful_iterations = p->chromosome_length
             max_unsuccessful_iterations =
                      i->memeplex[x].HCMethod_NoOfTimes
             CurrentHillClimbingMethod =
                      i->memeplex[x].HCMethod_ID
            while(current_successful_iterations
                   < max_successful_iterations and
                   current_unsuccessful_iterations
                   <max_unsuccessful_iterations) do
            begin
                temp_i = ApplyCurrentHillClimber(p,
                                    whole_chromosome,
                                    CurrentHillClimbingMethod,
                                    temp_i);
                 EvaluateIndividual(temp_i);
                 if( Better(temp_i->fitness, previous_fitness)) then
                 begin
                             CopyIndividual(temp_i, i);
                             prev_fitness := temp_i->fitness_value;
                             current_successful_iterations++;
                 end
                 else
                             //Current application of hillclimber may have
                             //corrupted the individual, so take its moves
                             //back and acquire the content of  the
                             //individual that has been achieved after the last
                             //successful modification
                            CopyIndividual(i, temp_i);
                            current_unsuccessful_iterations++;
                end
             end while
        end for
end
```

CHAPTER 9

RANDOM DATA GENERATION FOR PRIVATE SCHOOL TIMETABLING

This chapter explains the random data generator implemented to obtain synthetic problem instances for the private school timetabling.

9.1 Overview of a Possible Random Data Generation Method

Corne et. al. implemented a random problem generator for university exam timetabling. This program initially creates a random solution that contains a predefined number of exams. In the solution, all the exams are randomly assigned to available time slots. After that, all the constraints that are satisfied in the created solution instance are defined. In other words, if two exams are assigned different time slots in the solution, an edge constraint is defined between them. Such edge constraints are defined for all non-conflicting exams so that the set of edge constraints make the solution at hand the only solution. Then, some of those constraints are probabilistically removed to increase the number of possible solutions for the current problem. So, the problem instance becomes ready. In this way, it is assured that the generated problem has at least one solution. A similar approach can be employed to create syntatic problems for the private school timetabling.

9.2 The Process of Generating Random Data for Private School Timetabling

The below subsections discuss the creation of random data in detail. This process involves the determination of main aspects, namely the global

curriculum, problem size, temporal structure, event assignments and finally the constraints.

9.2.1 Creation of a Global Curriculum

During interviews with several private school authorities and instructors, it has been revealed that a private school has one commonly used curriculum. This curriculum is followed in all the branches belonging to the private school. It specifies the number of possible grades that a branch can offer, number of divisions available in each such grade and information about courses in the curriculum of each division defined. For instance, grades in a private school can be defined as Lycee 1, Lycee 2 and Lycee 3, while divisional choices for Lycee 1 can be the quantitative division or the verbal division. The students from each grade and division, i.e Lycee 2 verbal division students, have the same curriculum in all the branches of the private school. In the random data generator (RDG) for private school, initially a curriculum is constructed.

Each choice while generating the curriculum, i.e deciding how many grades will be available or how many courses will be assigned to a specific division, is made by choosing random values between realistic minimum and maximum parameters. In this way, statistically sound assignments can be achieved. Employing logically defined ranges in the form of maximum and minimum parameters for determining the numbers in the problem instance is applied throughout the RDG.

The curriculum also defines how many courses there will be for a specific division. It determines the number and length of meetings, the division that offers the course and the course name for each of those courses. Students from a particular division can be assigned to courses offered from other divisions as well. To illustrate, Lycee 2 students from verbal division usually take natural science courses, which are offered by the quantitative division. However, the number and length of meetings for the corresponding course sections are rather decreased since very low

coefficients are used when calculating exam scores belonging to questions on natural sciences for students from verbal division.

9.2.2 Determination of Problem Size

After the curriculum has been established in the RDG, values that specify the size of the problem instance are chosen. These values are listed below:

1. Number of branches
2. Number of grades in each branch
3. Number of divisions in each grade
4. Number of sections, i.e classes, in each division

Thus, the problem instance represents a hierarchical organization. Branches are divided into grades, grades are divided into divisions and finally divisions are divided into many sections, i.e classes. The size of the problem is determined by the listed values above and the curriculum. The number of course sections for each section, the meetings of those course sections are all determined according to the curriculum. As an event to be assigned in terms of private school timetabling is indeed a course section meeting, the size of the problem instance increases with increasing number of sections and course sections assigned to them as stated in the curriculum.

9.2.3 Definition for Temporal Structure

Each branch defines available time slots for each of its grades. Thus, a section belonging to a specific grade of the branch uses set of the available time slots for its grade. Although this set is same for all the sections of a specific grade, each section is assigned to slots from a subset of it. For instance, in a possible private school program, Lycee 2 students may attend to classes in the evenings, whereas Lycee graduates come to the private school at the weekends. In addition, some sections of

Lycee 2 students(or seventh grade students) may be attending to lectures from 6 to 8 pm, while some other sections of Lycee 2 students attend to lectures from 7 to 9 pm. Private schools generally provide such options for the students.

9.2.4 Slot and Instructor Assignments for the Events

Once the curriculum, the assignment of values representing the size of the problem and the temporal structure is ready, course sections for each of the sections are assigned to time slots. Meanwhile, the information on those course sections is obtained from the curriculum. After all the meetings for a course section has been assigned to time slots, an instructor is assigned to that course section. This instructor can be either a newly generated instructor or a previously generated instructor whose course sections don't have conflicts with the currently assigned course section. So, instructors are generated as necessary course sections for sections are created. After this process, number of instructors employed and total number of course sections in the problem instance is available. In addition, a possible solution for the generated problem instance that involves all the instructor and time assignments has been created. Furthermore, this solution involves some constraints already defined in it. This will become more clear in the following section.

9.2.5 Creation of Constraints

In the problem instance, some constraints are satisfied while the possible solution is being built, while some others are generated once the solution has been created. The method of solution generation with respect to certain constraints is not used in the sample random data generator of Corne et. al. They randomly created the possible solution and defined edge constraints thereafter. This approach doesn't appear to be efficient for the private school random data generation since the number of different constraint types is much larger.

The constraints defined in the problem instance along with the approach for generating them is listed in the next subsections.

9.2.5.1 Unary Event Constraints

1)Meetings *of course-sections can be assigned to predefined hours in predefined days(CS_PRE):* There is a parameter in the RDG that specifies the probability that a course section predefines time slots for its meetings. After the solution has been created, a subset of course sections are chosen probabilistically and their assigned slots in the solution instance are given as preset time slots for the problem instance.

2) Meetings *of course-sections should be assigned to allowable hours of the corresponding sections(S_PRE, S_EXC):* While each grade of a branch is generated, allowable time slots for this grade are also generated. During time assignments for course sections of a section that belongs to a specific grade, time slots to assign to meetings are chosen among the set of allowed time slots defined for the section's grade. Furthermore, there are several parameters in the RDG that specify the probabilities that a section excludes some time slots. Hence, a subset of sections are chosen probabilistically and some slots that are available for their grade but that are not used by the section in the solution at hand are given as excluded slots for that section.

9.2.5.2 Binary Event Constraints

1) Each section is assigned to at most one course-section at a given time period(EC1). While assigning a course section meeting for a section during the creation of the possible solution, a time slot that has not been used in the assignments for the events of the current section is chosen. Therefore, all constraints of this type are satisfied during the creation of the possible solution.

2) Each instructor is assigned to at most one course-section at a given time period(EC2). While assigning an instructor for a course section

during the creation of the possible solution, all the previously generated instructors are checked. If there is no instructor whose assigned course section meetings do not conflict with those of the current course section, a new instructor is generated and the total number of instructors increases by one. Therefore, all constraints of this type are satisfied in the possible solution.

3) Each classroom is assigned to at most one course-section at a given time period(EC3). The assignments of course sections to classrooms are not considered in the RDG since classroom assignments tend to be rather trivial due to sufficient allocation facilities in the private school timetabling problem.

9.2.5.3 Event-spread Constraints

1) *Each meeting of a course-section should be assigned to different days in a week (ES1).* While assigning time slots for each of the meetings belonging to a course section, it is checked that this constraint is satisfied.

2) An even distribution of verbal and quantitative courses should be achieved by assigning minimum and maximum hours for each course type in a day(S_DIVMINWL, S_DIVMAXWL). After the solution has been generated, minimum and maximum hours for courses offered from each division in the solution is calculated for a subset of sections which are probabilistically chosen. Then, the calculated values for those sections give the constaints for this constraint type.

3) There should be a minimum number of gaps between course-sections assigned to a section in a day unless stated otherwise.(i.e there can be a one-hour lunch break.) Hence, compactness of the daily timetable for a section should be achieved(S_MINGAP). Meetings of course sections for a section are not assigned to time slots in a way that satisfies this constraint. Therefore, there may be gaps in timetables for sections in the created solution instance.

4) Course-sections assigned to an instructor in a day should be consecutive and should contain a minimum number of gaps. This

constraint is necessary so that instructors who are paid for each lecture hour can finish their work at the private school *in a shorter period. So, compactness of the instructor daily timetables can also be required(I_MINGAP).* This constraint is implemented neither while generating the solution nor once the solution has been constructed. Minimizing the number of gaps for instructors and sections is not considered in the sample solution of the RDG but is dealt with in the penalty-allocating fitness function of the proposed framework. Therefore, the memetic algorithm implemented attempts to find a timetable with minimum number of gaps on instructors' timetables(and on sections' timetables as well).

5) Number of hours of course-sections assigned to a section in a day should be within predetermined minimum and maximum hours(S_MINWL, S_MAXWL). After the solution has been generated, minimum and maximum values for daily assigned hours in the solution for a subset of sections are calculated. Those sections are probabilistically chosen. Then, the calculated values for those sections give the constaints for this constraint type.

6) Courses requiring more intellectual activity should be placed earlier in the timetables than the ones that are generally accepted as being rather easier(ES6). The type of each course section is determined by the division that offers it. Constraints of this type are not dealt with in RDG or the proposed framework.

9.2.5.4 Instructor Constraints

1) The specifications involving allowable hours of instructors should be satisfied (I_EXC). After the solution has been generated, a subset of instructors are chosen probabilistically. For each such instructor, several time slots which remain unassigned for them in the solution are again stochastically chosen and given as excluded time slots for the instructor.

2) *Since a* private school *can have several faculties in different buildings, travelling times of instructors between these buildings should be minimized(I_MAXLOC).* While assigning instructors to course sections, it

is always satisfied that an instructor gives lectures at maximum two branches on a day. Since the course section assignments for branches are performed in order, this constraint is already satisfied in the generated solution.

3) *Number of hours of course-sections assigned to an instructor in a day should also be within predetermined minimum and maximum hours(I_MINWL, I_MAXWL).* After the solution has been generated, a subset of instructors are chosen probabilistically. Their minimum and maximum values for daily assigned hours in the solution is calculated. Then, the calculated values for those instructors give the constaints for this constraint type.

4) *Travelling times of instructors should also be considered when assigning course-sections to them in a day(I_TRAVEL)*. During instructor assignments for creating the solution instance, it is satisfied that there is at least one hour travelling time left between two course-sections that will be offered to sections of different branches in different locations by the same instructor.

9.3 Assumptions for the RDG

Default assumptions for the OSS program of a private school are listed below:

1) A problem instance has a predefined number of branches, grades, divisions and sections. Each branch has several grades. Each of those grades are divided into divisions. Finally, each such division has several sections, i.e classes.

2) All sections within the same division and grade, i.e all Lycee 2 students from the quantitative division, have the same curriculum. Therefore, a global curriculum, which determines which courses a section should attend to according to its division and grade, should be followed in all branches of the private school. This curriculum keeps the number and lengths of meetings for the courses as well.

3) The division number and grade number pair of a section uniquely identifies with courses the section should take. Sections from a division may be assigned to courses from other divisions as well. To illlustate, a section of the verbal division can have courses from the quantitative division. This fact results from the score calculation of the OSS examination.

4) Each grade has a set of available time slots. These time slot assignments for the grades may vary from branch to branch. However, all the sections belonging to a specific grade in a specific branch have the same set of available time slots. Besides, not all such sections may use the same time slot subset from this set. Therefore, timetables of different sections in the same grade may differ slightly according to their exclusion constraints defined.

5) Each instructor is assigned to a number of course sections. Each meeting of the same course-section is always taught by the same instructor.

These assumptions readily apply to different programs of the private schools other than the OSS program. They stem from the interviews and investigations on the private school in Turkey.

9.4 Parameters for the RDG

There are several parameters such as number of time slots, number of instructors, etc. to be assigned for the problem generation. All the parameters are assigned values that are suitable for private school timetabling. Those values are randomly selected among a range of statistically appropriate choices. Therefore, minimum and maximum limits for each such parameter is given to the random data generator. Those parameters are listed in the appendix.

9.5 Pseudocode for the RDG

The pseudocode for the RDG is as displayed in Table 9.1. The RDG code creates a synthetic problem instance and solves it by satisfying all the edge constraints defined in the problem. Then, it defines several additional constraints on the problem as stated before. Below is a summary.

1) Preset constraints for course sections, exclusions for sections and instructors, event-spread constraints about the daily workloads of instructors and sections are set once the random solution has been created.
2) Other constraints(edge constraints, other event-spread constraints and other instructor constraints) are satisfied owing to heuristics employed during time slot and instructor assignments.

9.6 Output of the RDG

The output of the RDG consists of an input file where the problem is written with its constraints, a file containing a sample solution and a file containing the analysis of the created problem instance. A sample input and analysis file can be found in the appendix.

Table 9.1 The algorithm for RDG

```
//First generate the global curriculum
Generate available grades
For each grade available
        Generate available divisions
        For each division available
                Choose available courses from own division
                courses := courses  U courses from other divisions
                For each course
                        Generate number and length of meetings
//Generate the sections and
//Assign the necessary course sections to a generated section
Generate a number of branches
For each branch
        Choose a number of grades
        For each grade
                Generate available time slots
                Choose a number of divisions
                For each division
                        Generate a number of sections
                        For each section
                                Create course sections according to the curriculum
                                For each course section
                                        Assign available time slots of its grade to
                                                each of the meetings of course section
                                        Assign an instructor available to course
section
//Once the problem is set and its solution has been created
 Generate exclusion constraints for some instructors and sections among their
 unused time slots in the solution
 Generate workload constraints for some instructors and sections
```

CHAPTER 10

EXPERIMENTS AND DISCUSSION

10.1 Test Data

Eight different private school problem instances have been produced by the implemented random data generator. Table 10.1 displays the analysis of each of those test cases. In this table, the total number of events, sections, divisions, grades, instructors, etc. are all listed for each problem instance. In addition, the average number(ρ) and maximum number of total workload for instructors and sections are displayed in the table. The values for conflict densities of edge constraints in test cases, denoted as CF in the table, are rather small. However, this does not imply that the problem instances are easy to solve. As explained in section 2.5.5, the conflict density for any private school timetabling problem where more than 10 sections and instructors are involved cannot be greater than 0.2. Therefore, achieving a minimum number of gaps, i.e empty slots between lecture hours for students and instructors, becomes the constraint that is most difficult to satisfy. Besides, none of the test cases experimented with in this study have a feasible timetable solution with no gaps for all the instructors and students involved. For all the problem instances, there are 8 days and 10 daily hours in the timetable. In each of these problems, percent of course sections that define preset slots is about %5. The problem instances presented in Table 10.1 each have different properties, i.e different number of branches, instructors or work loads. Memetic algorithm employing violation-directed hierarchical hill climbing and newly proposed genetic operators, pure genetic algorithm and multimeme algorithms are experimented on the randomly generated test cases. The following sections present and discuss the

Table 10.1 Analysis of Test Data. ρ refers to the average occupancy rate(Alkan et. al. 2003), which means the average number of assigned hours for instructors or sections.

	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8
Meetings	186	312	232	108	378	188	408	438
Course sections	39	62	99	18	66	51	83	88
Sections	5	14	22	4	9	7	9	18
Divisions	5	14	22	2	6	6	9	18
Grades	2	6	9	2	6	5	3	6
Branches	1	2	3	1	2	3	1	2
Instructors	12	17	13	8	21	14	17	21
CF	0.26	0.12	0.11	0.33	0.14	0.19	0.16	0.09
ρ (sections)	47.20	27.28	13.18	35.00	50.66	32.42	57.11	29.77
ρ (instructors)	19.66	22.47	22.30	17.50	21.71	16.21	30.24	25.52
Max. section	58	32	16	39	52	34	67	40
Max. instructor	34	33	33	25	32	23	41	41
Percent of sections defining workload constraints	80.00	57.14	50.00	50.00	66.66	57.14	33.33	33.33
Percent of instructors defining workload constraints	66.66	41.17	61.53	62.50	42.85	50.00	41.17	47.61

results for those algorithms and aim to figure out the algorithm with best performance on the private school timetabling problem instances.

10.2 Experimental Settings

There are 3 sets of experiments performed in this study. The first set of experiments were done to reveal the performance of the proposed memetic algorithm. In this set, experiments were carried out to figure out

1. the combination of which proposed crossover and mutation operators perform the best with the implemented VDHC method,
2. the success of VDHC method without using the crossover operator,
3. the success of the 12 different low-level hill climbers without the management of the VDHC method.

Once these experiments were accomplished, the results for the VDHC method along with the best combination of crossover and mutation operators among the set of newly proposed operators were compared with those for pure genetic algorithms and multimeme algorithms in the next sets of experiments.

The second set of experiments aims to reveal the performance of the pure genetic algorithm on the problem instances. In order to compare traditional genetic algorithm and the proposed memetic algorithm appropriately, either maximum number of generations allowed or the population size for the genetic algorithm must be more than those for the memetic algorithm. If both algorithms are processed within the same number of maximum generations allowed and the same population size, an unfair comparison arises. The memetic algorithm searches states, i.e different individuals or possible solutions, at least as many as maximum number of unsuccessful hill climbing steps, say N steps, during a local search phase for an individual. So, if there are P individuals for the memetic algorithm with transgenerational replacement strategy, there

will be at least N(P-2) states evaluated in each generation. However, the genetic algorithm with no local optimization phase searches P-2 states in each generation. As a result, keeping the number of generations for the two algorithms constant and comparing best solutions thereafter would not lead to a successful comparison in this case. Instead, it should be maintained that the average number of evaluations per run for the genetic algorithm must be at least as much as the number of evaluations for the memetic algorithm. The fitness evaluation is the most time-consuming process for evolutionary algorithms. Moreover, the memetic algorithm implementation utilizes fast fitness evolution after each hill climbing step by only considering the violations in the modified regions of the chromosome. So, if the genetic algorithm performs evaluations at least as many as the memetic algorithm, it will be allowed to execute at least as long as the memetic algorithm and will have enough chance to exhibit its performance.

In the third set of experiments, the multimeme algorithm is applied. In the first phase of these experiments, the individuals in the population each carry merely one meme that identifies the hill climber to apply to the individual. In the second phase, again multimeme approach is applied but this time an individual carries memes as many as the number of different hill climbers. In one local search step, those hill climbers are applied to the individual in the order their corresponding memes are located in the individual's memeplex.

Transgenerational replacement strategy and tournament selection method with a tour size of 4 for mate selection were employed in the experiments. Maximum number of unsuccessful hill climbing steps during the local optimization phase of an individual is kept as 10. The next section presents and discusses the results of each set of experiments.

10.3 Results

10.3.1 Results for Memetic Algorithms

The first set of experiments aims to determine the combination of which crossover and mutation operators among the set of newly proposed operators gives the best results for the data instances. These experiments are performed in two steps. In the first step, the best mutation operator is determined and it is used while identifying the best crossover operator in the second step. Then experiments without utilizing a crossover operator and experiments without the VDHC method are performed to indicate the contribution of each of those methods to the memetic algorithm.

Unless otherwise stated, an experiment in the first set of experiments consists of 30 runs each with 2000 generations by utilizing a specific crossover and mutation operator and the VDHC method is kept on. Population size is 50 for test cases from 1 to 6 and 100 for the remaining cases. In the first-step experiments, all the 16 newly introduced mutation operators are tested both with traditional uniform crossover and traditional one-point crossover operators. From the studies of Alkan et. al. (2003) and Ozcan et. al. (2005a), it is known that traditional crossover and mutation operators perform at least as well as the genetic operators designed to act on a certain level of the hierarchy on the chromosome rather than the whole chromosome for several timetabling problems. Therefore, in the first step of the initial experiments, both uniform and one-point crossover operators are used while testing all the new mutation operators. So, 32 different experiments are performed for each data instance in the first step of this set of experiments. The duration of each experiment varies approximately between 70 minutes to 6,5 hours according to the size of the problem instance that is being tested. Therefore, a duration varying between 1,5 and 8,6 days is necessary to perform all the first-step experiments on a data instance. In order to shorten this duration, about 40 computers with Windows ME

Table 10.2 Results for new mutation operators when used with traditional crossover operators

	Test1			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	29.80	0	30.00	0
RAND_SECTION_MT	**27.16**	**0**	29.33	0
RAND_DIVISION_MT	30.60	0	30.66	0
RAND_GRADE_MT	28.60	0	30.93	0
RAND_BRANCH_MT	28.86	0	29.90	0
RAND_INSTRUCTOR_MT	31.40	0	34.96	0
ALL_SECTION_MT	198.80	0.2	177.06	0
ALL_DIVISION_MT	208.76	0.1	173.76	0
ALL_GRADE_MT	71.83	0	51.80	0
ALL_BRANCH_MT	28.80	0	32.40	0
ALL_INSTRUCTOR_MT	239.86	3.53	253.13	3.13
VD_SECTION_MT	**27.16**	**0**	29.46	0
VD_DIVISION_MT	**27.56**	**0**	**28.23**	**0**
VD_GRADE_MT	27.96	0	31.06	0
VD_BRANCH_MT	28.73	0	**28.36**	**0**
VD_INSTRUCTOR_MT	27.86	0	**26.63**	**0**

	Test2			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	101.73	0	111.10	0
RAND_SECTION_MT	**98.66**	**0**	**104.76**	**0**
RAND_DIVISION_MT	104.53	0	107.40	0
RAND_GRADE_MT	**97.13**	**0**	111.20	0
RAND_BRANCH_MT	100.36	0	110.56	0
RAND_INSTRUCTOR_MT	104.70	0	109.96	0
ALL_SECTION_MT	584.10	11.4	574.80	10.56
ALL_DIVISION_MT	586.33	11.9	598.96	10.46
ALL_GRADE_MT	523.70	2.43	490.10	1.03
ALL_BRANCH_MT	166.73	0	152.10	0
ALL_INSTRUCTOR_MT	573.70	11.63	592.36	11.33
VD_SECTION_MT	100.00	0	108.90	0
VD_DIVISION_MT	99.43	0	**103.80**	**0**
VD_GRADE_MT	101.83	0	**106.96**	**0**
VD_BRANCH_MT	99.30	0	111.60	0
VD_INSTRUCTOR_MT	**96.56**	**0**	107.43	0

Table 10.2 (cont'd) Results for new mutation operators when used with traditional crossover operators

	Test 3			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	22.86	0	24.20	0
RAND_SECTION_MT	23.03	0	23.66	0
RAND_DIVISION_MT	23.06	0	24.20	0
RAND_GRADE_MT	24.06	0	23.50	0
RAND_BRANCH_MT	24.13	0	24.03	0
RAND_INSTRUCTOR_MT	26.50	0	26.96	0
ALL_SECTION_MT	374.60	3.13	360.53	2.93
ALL_DIVISION_MT	374.60	3.13	382.36	3.06
ALL_GRADE_MT	345.56	0.30	305.66	0
ALL_BRANCH_MT	101.0	0	74.10	0
ALL_INSTRUCTOR_MT	374.50	1.33	353.23	0.36
VD_SECTION_MT	**22.66**	**0**	23.73	0
VD_DIVISION_MT	**22.66**	**0**	**22.36**	**0**
VD_GRADE_MT	**22.16**	**0**	23.30	0
VD_BRANCH_MT	23.03	0	**22.90**	**0**
VD_INSTRUCTOR_MT	**22.03**	**0**	**23.23**	**0**

	Test 4			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	4.80	0	4.53	0
RAND_SECTION_MT	4.76	0	5.03	0
RAND_DIVISION_MT	4.93	0	**3.26**	**0**
RAND_GRADE_MT	4.03	0	4.73	0
RAND_BRANCH_MT	4.30	0	4.30	0
RAND_INSTRUCTOR_MT	7.60	0	4.96	0
ALL_SECTION_MT	56.06	0	44.33	0
ALL_DIVISION_MT	22.76	0	12.03	0
ALL_GRADE_MT	22.10	0	16.00	0
ALL_BRANCH_MT	4.16	0	**3.86**	**0**
ALL_INSTRUCTOR_MT	86.46	0	83.70	0
VD_SECTION_MT	**3.50**	**0**	**3.60**	**0**
VD_DIVISION_MT	5.20	0	5.63	0
VD_GRADE_MT	**2.96**	**0**	5.46	0
VD_BRANCH_MT	**4.00**	**0**	4.86	0
VD_INSTRUCTOR_MT	5.36	0	4.06	0

Table 10.2 (cont'd) Results for new mutation operators when used with traditional crossover operators

	Test 5			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	**168.93**	0	183.90	0
RAND_SECTION_MT	170.76	0	182.20	0
RAND_DIVISION_MT	172.53	0	191.76	0
RAND_GRADE_MT	175.86	0	186.73	0
RAND_BRANCH_MT	171.76	0	182.43	0
RAND_INSTRUCTOR_MT	183.63	0	184.46	0
ALL_SECTION_MT	500.63	22.26	508.16	18.50
ALL_DIVISION_MT	498.20	14.66	491.33	8.86
ALL_GRADE_MT	504.06	14.70	493.36	8.73
ALL_BRANCH_MT	289.23	0	244.56	0
ALL_INSTRUCTOR_MT	524.56	27.23	511.43	26.43
VD_SECTION_MT	170.16	0	**181.56**	**0**
VD_DIVISION_MT	**169.80**	**0**	**178.,63**	**0**
VD_GRADE_MT	**168.63**	0	**179.43**	**0**
VD_BRANCH_MT	170.86	0	187.53	0
VD_INSTRUCTOR_MT	172.76	0	183.53	0

	Test 6			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	58.00	0	62.43	0
RAND_SECTION_MT	60.70	0	59.36	0
RAND_DIVISION_MT	56.10	0	62.10	0
RAND_GRADE_MT	56.63	0	59.30	0
RAND_BRANCH_MT	55.66	0	59.20	0
RAND_INSTRUCTOR_MT	62.70	0	63.30	0
ALL_SECTION_MT	315.53	3.33	310.40	1.56
ALL_DIVISION_MT	298.16	1.90	277.90	0.73
ALL_GRADE_MT	270.93	0.63	231.83	0.1
ALL_BRANCH_MT	159.06	0	132.40	0
ALL_INSTRUCTOR_MT	337.76	8.40	324.96	7.33
VD_SECTION_MT	**54.36**	**0**	**55.83**	**0**
VD_DIVISION_MT	**55.16**	**0**	58.23	0
VD_GRADE_MT	**53.00**	**0**	60.53	0
VD_BRANCH_MT	56.53	0	**56.60**	**0**
VD_INSTRUCTOR_MT	56.80	0	**56.53**	**0**

Table 10.2 (cont'd) Results for new mutation operators when used with traditional crossover operators

	Test 7			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	128.66	0	137.33	0
RAND_SECTION_MT	132.16	0	138.10	0
RAND_DIVISION_MT	**123.36**	**0**	**132.40**	**0**
RAND_GRADE_MT	126.26	0	135.63	0
RAND_BRANCH_MT	129.73	0	136.10	0
RAND_INSTRUCTOR_MT	133.10	0	136.86	0
ALL_SECTION_MT	430.20	34.60	434.93	28.56
ALL_DIVISION_MT	434.83	34.40	436.93	29.06
ALL_GRADE_MT	409.43	9.93	305.86	0.13
ALL_BRANCH_MT	129.50	0	137.33	0
ALL_INSTRUCTOR_MT	436.10	40.26	451.30	37.26
VD_SECTION_MT	**124.00**	**0**	**133.26**	**0**
VD_DIVISION_MT	126.93	0	133.73	0
VD_GRADE_MT	126.70	0	138.03	0
VD_BRANCH_MT	126.50	0	138.40	0
VD_INSTRUCTOR_MT	**124.70**	**0**	**133.00**	**0**

	Test 8			
	UX		1PTX	
	σ	β	σ	β
COURSESECTION_MT	155.46	0	164.93	0
RAND_SECTION_MT	154.16	0	167.46	0
RAND_DIVISION_MT	158.76	0	**163.50**	**0**
RAND_GRADE_MT	150.73	0	167.30	0
RAND_BRANCH_MT	154.16	0	167.90	0
RAND_INSTRUCTOR_MT	165.30	0	174.16	0
ALL_SECTION_MT	793.53	26.36	776.10	25.4
ALL_DIVISION_MT	791.53	25.9	776.10	25.4
ALL_GRADE_MT	750.76	13.80	728.63	4.73
ALL_BRANCH_MT	318.06	0	248.90	0
ALL_INSTRUCTOR_MT	793.86	27	794.23	25.46
VD_SECTION_MT	**148.06**	**0**	**152.36**	**0**
VD_DIVISION_MT	148.36	0	163.73	0
VD_GRADE_MT	**146.13**	**0**	170.43	0
VD_BRANCH_MT	158.80	0	164.16	0
VD_INSTRUCTOR_MT	**148.20**	**0**	**162.86**	**0**

installed and 256MB RAM utilized were used for the first and second step experiments. Thus, different experiments for the same problem instance were spread-out on different computers of equal capacity.

The long duration of the first-step experiments resulted in using rather small-sized test data instances for the experiments. However, these instances have different properties, i.e different number of branches or conflict densities, etc. So, the results are expected to adequately reflect the performance of the newly proposed global search operators.

Table 10.2 shows the results for all the mutation operators both with uniform crossover(UX) and one-point crossover(1PTX) on the eight test cases. The symbols σ and β denote the average number of soft and hard constraint violations in the best individual found after 2000 generations. All the results in the table are averaged over 30 runs. Results in bold indicate the top three configurations for each test instance. From these results, it can be inferred that the traditional mutation operator, namely COURSESECTION_MT, performs not far worse than the newly proposed operators. Indeed, it performs much better than the type of the new mutation operators that apply traditional mutation to all of their corresponding classifiers. These operators are far more destructive. To illustrate, ALL_INSTRUCTOR_MT, which applies traditional mutation to the genes of each instructor, can be examined. This operator randomly chooses a course section of each instructor and mutates it. It appears as the most destructive and also the worst-performing operator among the set of the mutation operators.

Another type of the mutation operators, which randomly choose their corresponding classifier and apply traditional mutation on that part of the chromosome, are approximately as good as the traditional mutation operator for many test cases. Besides, violation-directed mutation operators seem to be more promising than the other types among the newly introduced mutation operators. A violation-directed mutation operator chooses its corresponding classifier with tournament selection

and randomly mutates a gene in the chosen classifier. In the tournament selection, a classifier's chance of being selected increases with the number of constraint violations contained in the subsolution it represents. When the results for random mutations and violation-directed mutations are compared, it can be claimed that the utilization of tournament selection for choosing a classifier slightly improves the performance of the mutation operator for most of the test cases. Test 1, 4 and 7 contain merely one branch in their private school organization. Therefore, COURSESECTION_MT, RAND_BRANCH_MT and ALL_BRANCH_MT apply the same operation on the chromosome for those instances. Since there is one branch for those instances, RAND_BRANCH_MT and ALL_BRANCH_MT both act on the whole chromosome just as COUSESECTION_MT, i.e the traditional mutation operator that mutates a gene, which is a course section including all its meetings in the proposed framework. Corresponding results in table 10.2 indicate that the number of soft constraint violations reached for these three types of mutation operators are very similar for test cases 1, 4 and 7.

Mutation operators that apply traditional mutation to all of the corresponding classifiers in the chromosome have the worst performance. Their implementation appears rather useless but it reflects the fact that the degree of modification for mutation must be limited not to corrupt the individual that has already been improved to some extent by global and local search. Traditional mutation operator, random boundary mutations, i.e mutation operators that randomly choose a classifier to mutate one of its genes randomly, and violation-directed mutations have very similar outcomes. In detail, violation-directed mutations appear as slightly better mutation operators when the overall performance of the operators for all the instances are compared.

Figure 10.1 compares the best individuals in each generation for VD_GRADE_MT, RAND_BRANCH_MT, traditional mutation (or COURSESECTION_MT) and ALL_BRANCH_MT on test 6 data instance.

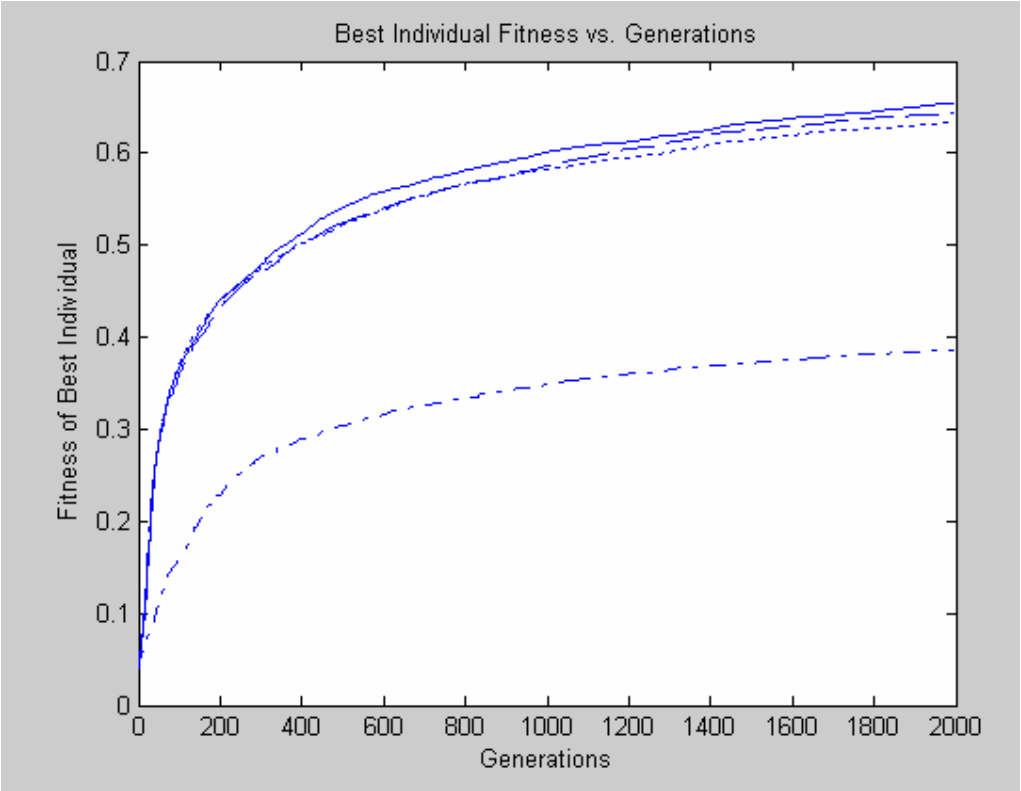

Figure 10.1 Comparison of best mutation operators among their group for test case 6. Solid, dashed, dotted and dashed-dotted lines represent results for VD_GRADE_MT, RAND_BRANCH_MT, COURSESECTION_MT and ALL_BRANCH_MT respectively.

These are the best operators among their corresponding group of mutations for this problem. For instance, VD_GRADE_MT has the best result among the other violation-directed mutation operators for this test case. Similarly, RAND_BRANCH_MT gives the least average soft constraint violations among all the mutation operators that choose a random classifier and mutates one of its gene.

ALL_BRANCH_MT appears as the least destructive and thus most successful mutation among the set of operators that apply traditional mutation to every classifier they correspond to. As Figure 10.1 illustrates, VD_GRADE_MT, whose results are drawn with solid lines, performs slightly better than RAND_BRANCH_MT and traditional mutation. Generally, violation-directed mutations have slightly better performance. Hence, the second-step experiments are performed by using one of them, namely VD_GRADE_MT, as the mutation operator. VD_GRADE_MT selects a grade that has more constraint violations on the chromosome and mutates a randomly chosen gene in the selected grade. In other words, it acts as if the selected grade were the whole chromosome at hand. 13 different crossover operators are each tested on the eight data instances. During these second-step experiments, again memetic algorithm that employs VDHC is employed. The results that are averaged over 30 runs are displayed in table 10.3. Again, σ and β denote the average number of soft and hard constraint violations found in the best individual after 2000 generations. All the results in the table are averaged over 30 runs. The results in table 10.3 show that uniform crossover generally perform better than the newly proposed operators. The new operators can only compete with traditional one-point crossover operator.

The uniform crossover operators that treat each of their corresponding classifiers as a gene, i.e the operators that end with the UX suffix, are not good alternatives for the traditional uniform crossover. Therefore, it

Table 10.3 Results for new crossover operators when used with VD_GRADE_MT mutation operator

Crossover Operator		Test 1		Test 2	
		σ	β	σ	β
SECTION_UX	14	**29.86**	**0**	108.46	0
DIVISION_UX	13	**30.20**	0	**98.10**	**0**
GRADE_UX	12	32.93	0	**103.76**	**0**
BRANCH_UX	11	X	X	113.86	0
I_UX	15	**30.40**	0	105.86	0
VD_COURSESECTION_1PTX	9	32.10	0	110.03	0
VD_SECTION_1PTX	8	31.63	0	104.96	0
VD_DIVISION_1PTX	7	33.60	0	109.80	0
VD_GRADE_1PTX	6	31.16	0	**104.76**	**0**
VD_BRANCH_1PTX	5	X	X	113.06	0
VD_I_1PTX	10	30.50	0	105.83	0

Crossover Operator	Test 3		Test 4	
	σ	β	σ	β
SECTION_UX	**22.43**	**0**	**3.90**	**0**
DIVISION_UX	23.33	0	4.53	0
GRADE_UX	**23.03**	**0**	5.00	0
BRANCH_UX	23.96	0	X	
I_UX	23.36	0	**4.33**	**0**
VD_COURSESECTION_1PTX	23.53	0	**4.50**	**0**
VD_SECTION_1PTX	24.06	0	5.16	0
VD_DIVISION_1PTX	24.80	0	4.93	0
VD_GRADE_1PTX	**22.80**	**0**	4.93	0
VD_BRANCH_1PTX	23.40	0	X	
VD_I_1PTX	23.86	0	4.80	0

Table 10.3 (cont'd) Results for new crossover operators when used with VD_GRADE_MT mutation operator

Crossover Operator	Test 5		Test 6	
	σ	β	σ	β
SECTION_UX	180.40	0	60.33	0
DIVISION_UX	**178.80**	**0**	59.10	0
GRADE_UX	181.16	0	60.86	0
BRANCH_UX	182.70	0	60.16	0
I_UX	**173.76**	**0**	61.43	0
VD_COURSESECTION_1PTX	**177.40**	**0**	59.63	0
VD_SECTION_1PTX	179.06	0	**58.36**	**0**
VD_DIVISION_1PTX	182.06	0	59.70	0
VD_GRADE_1PTX	185.66	0	**57.80**	**0**
VD_BRANCH_1PTX	183.00	0	**57.50**	**0**
VD_I_1PTX	183.36	0	59.76	0

Crossover Operator	Test 7		Test 8	
	σ	β	σ	β
SECTION_UX	**136.66**	**0**	**156.43**	**0**
DIVISION_UX	**133.06**	**0**	**156.43**	**0**
GRADE_UX	137.76	0	166.90	0
BRANCH_UX	X		174.20	0
I_UX	**130.60**	**0**	164.16	0
VD_COURSESECTION_1PTX	136.70	0	164.53	0
VD_SECTION_1PTX	138.40	0	165.46	0
VD_DIVISION_1PTX	136.90	0	169.46	0
VD_GRADE_1PTX	139.33	0	**163.23**	**0**
VD_BRANCH_1PTX	X		170.83	0
VD_I_1PTX	140.13	0	169.03	0

is best to keep the unit of inheritance as genes, or course sections, rather than larger groupings such as sections or grades, etc. The violation-directed one-point crossover operators choose their corresponding classifier among the ones with more constraint violations via tournament selection and take this classifier as the crossover point. To illustrate, VD_SECTION_1PTX selects a section with more constraint violations and assigns it as the crossover point during the recombination process. Unlike the case in mutation, this violation-directed selection process does not improve the results at all. BRANCH_UX and VD_BRANCH_1PTX cannot work on data instances with only one branch. Therefore, such cases are marked with an X symbol on table 10.3.

Table 10.4 displays the results of the memetic algorithm employing the VDHC method and traditional genetic operators for each data instance. These results are listed again along with the average number of evaluations per run to identify the success brought about by VDHC method and crossover. Tables 10.5 and 10.6 display the results of the best individual for the memetic algorithm when only crossover operation and only VDHC method are disabled respectively.

When crossover is not utilized, all the genes of a parent pass to one offspring and all the genes of the second parent pass to the other offspring. So, the genetic contents of the parents are directly copied to the offsprings. Results in table 10.5 indicate that disabling recombination process does not cause any hard constraint violations in the best individual. However, the number of soft constraint violations significantly increases for each of the test cases. Therefore, we can conclude that the recombination operation is still necessary while VDHC method coordinates the low-level hill climbers.

In order to state the performance improvement of the memetic algorithm brought about by the VDHC method, hill climbers are experimented on the

Table 10.4 Results for memetic algorithm utilizing uniform crossover and traditional mutation along with VDHC Method

Test Case	Avg. No. of Soft Constraint Violations	Avg. No. of Hard Constraint Violations	Avg. No. of Evaluations
Test 1	29.80	0	1,215,225.93
Test 2	101.73	0	1,181,180.50
Test 3	22.86	0	1,175,149.46
Test 4	4.80	0	1,230,088.16
Test 5	168.93	0	1,170,519.13
Test 6	58.00	0	1,177,595.96
Test 7	128.66	0	2,415,368.03
Test 8	155.46	0	2,396,330.70

Table 10.5 Results for memetic algorithm without crossover operator while still utilizing VDHC method

Test Case	Avg. No. of Soft Constraint Violations	Avg. No. of Hard Constraint Violations	Avg. No. of Evaluations
Test 1	33.66	0	1,199,002.03
Test 2	116.70	0	1,170,189.80
Test 3	27.43	0	1,161,541.46
Test 4	5.46	0	1,243,200.93
Test 5	193.53	0	1,157,900.46
Test 6	70.40	0	1,166,520.76
Test 7	144.50	0	2,365,706.96
Test 8	185.26	0	2,358,542.80

Table 10.6 Results for memetic algorithm without VDHC operator while traditional genetic operators are employed

Test Case	Avg. No. of Soft Constraint Violations	Avg. No. of Hard Constraint Violations	Avg. No. of Evaluations
Test 1	40.43	0	1,204,912,20
Test 2	220.90	0	1,170,424,70
Test 3	42.13	0	1,166,600,36
Test 4	9.80	0	1,254,634,60
Test 5	236.50	0	1,159,236,00
Test 6	81.53	0	1,180,109,90
Test 7	153.63	0	2,402,756,66
Test 8	283.36	0	2,402,954.86

test cases when VDHC method is not employed and the low-level hill climbers are randomly chosen in the local search phase. In this case, no level, i.e work area, is specified for any hill climber and they act on the whole chromosome.

Results in table 10.6 are clear indications for the necessity of the VDHC method since leaving the operation of hill climbers to randomness leads to much more soft constraint violations in each of the test cases.

Figure 10.2 displays the average best individual fitness in each generation for different configurations of the memetic algorithm, which are employing both VDHC method and traditional genetic operators, disabling only the crossover operation and disabling only the VDHC method respectively. The results belong to test case 8. Since tables 10.4, 10.5 and 10.6 show that similar graphs would be drawn for the rest of the test cases, only the results of one data instance have been represented as a graph for illustration purpuses.

As figure 10.2 displays, the utilization of the VDHC method greatly enhances the memetic algorithm. In addition, memetic algorithm is significantly enhanced by the application of recombination operation.

It can be argued that the comparison offered by tables 10.4, 10.5 and 10.6 is fair enough since the average number of evaluations per run for each of the three configurations discussed are close to each other for each of the test cases.

In an evolutionary algorithm, the fitness of the best individual must be observed as generations pass since it may stop increasing and may even start to decrease after certain number of generations. Thus, the stopping criteria for an evolutionary algorithm must also take this into account.

Figure 10.3 displays the fitness of best individual vs. generations for the memetic algorithm with VDHC method, uniform crossover and traditional

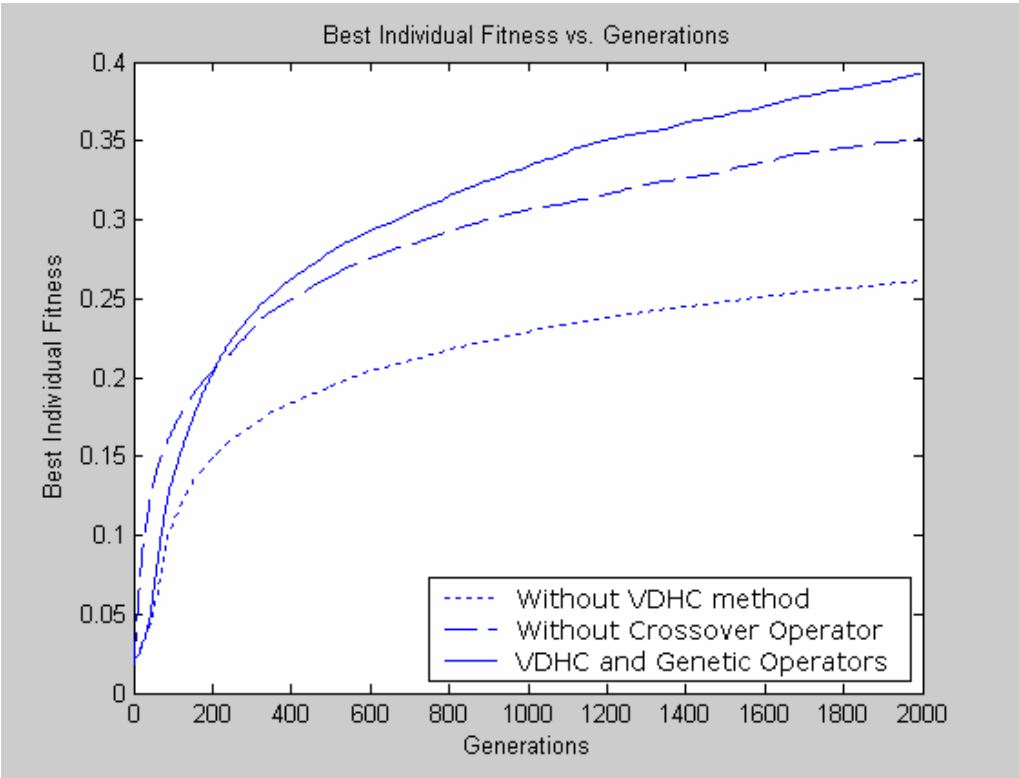

Figure 10.2 Comparison of results for different combinations of VDHC and crossover utilization on test case 8

mutation. Number of constraint violations of the best individual after 2000 generations resulted in these experiments are listed in table 10.4. Figure 10.3 reveals that only the best individuals for test case with the least number of events, namely test case 4, begin to have less fitness values even before 1000 generations. However, the fitness for best individuals continue increasing for the remaining test cases. So, using at least 2000 generations is necessary for them.

As figure 10.3 displays, problem instance of test case 4 is easiest to solve for the memetic algorithm among all the test cases although its conflict density of edge constraints is higher. Therefore, the complexity of a problem instance for private school timetabling is highly dependant upon the size of the data, i.e the number of meetings involved, rather than the intensity of edge constraints. The problem instances with larger sizes, test cases 5, 7 and 8, have all less fitness values for best individuals during each generation when compared with other data instances.

Figure 10.4 shows similar results for the memetic algorithm in the absence of the VDHC method. When VDHC method does not coordinate the process of low-level hill climbers, the overall best individual fitness values in each generation for the test cases significantly reduce. In this case, best individual fitness values for test case 4 merely continue improving as figure 10.4 illustrates. This shows that it will take more number of generations for the results of this test case to reach their maximum values and then begin falling as illustrated in figure 10.3.

As discussed in section 7.7, there are 12 different low-level hill climbers implemented to resolve the violations of a specific constraint type for the private school timetabling problem. The VDHC method determines which hill climber will be applied to which portion of the current individual's chromosome in the local search phase. The hill climbers are divided into two groups. The 6 hill climbers in the first group aim to resolve the constraint violations corresponding to sections, i.e students,

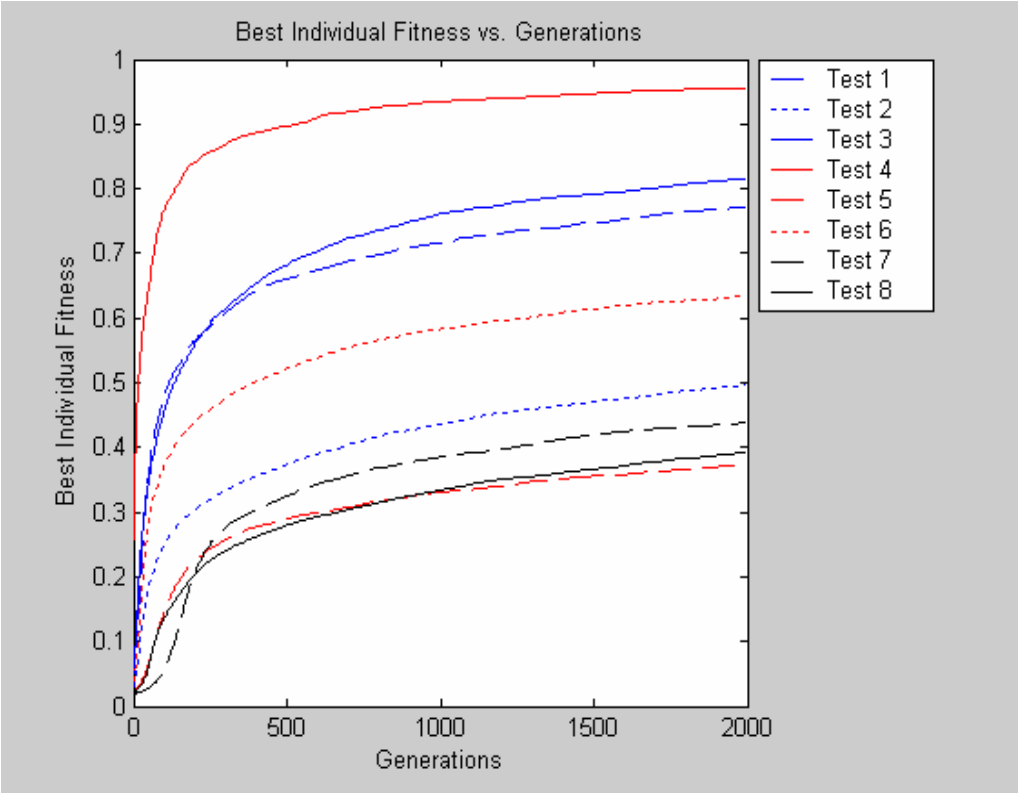

Figure 10.3 Best individual fitness in generations for the memetic algorithm with VDHC method and traditional genetic operators

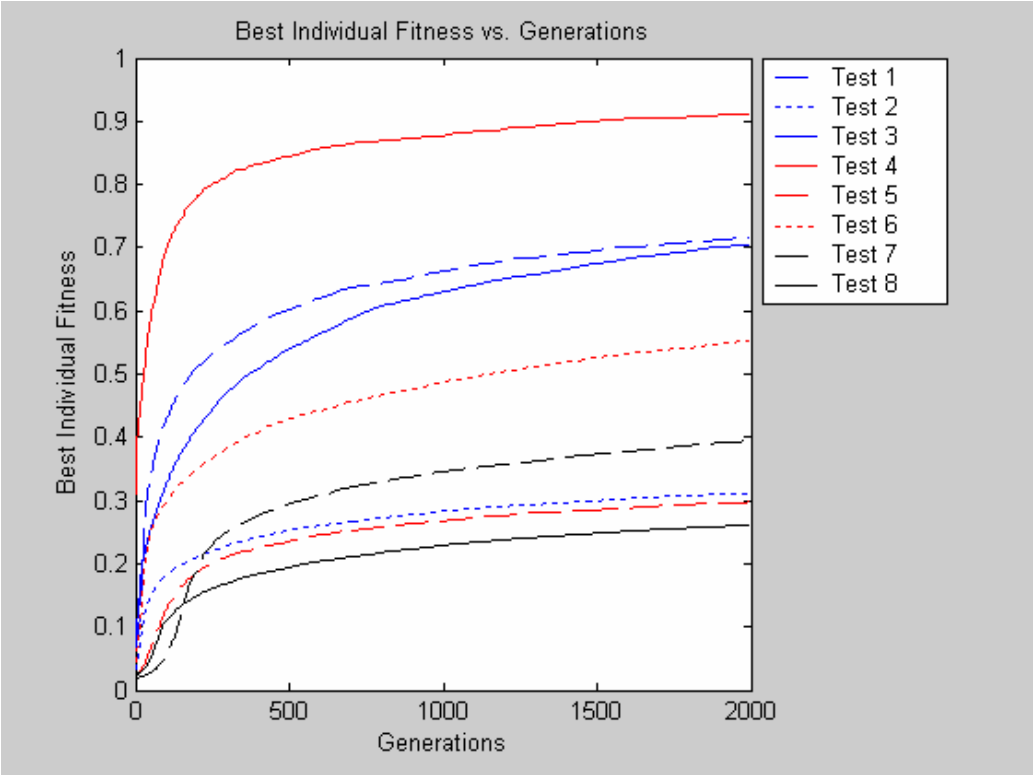

Figure 10.4 Best individual fitness in generations for the memetic algorithm employing traditional genetic operators without utilizing VDHC method

while the other 6 second group hill climbers attempt to resolve instructors' constraint violations. When the modification brought about by a hill climber improves the fitness of the individual, the success rate of the hill climber increases. However, if the hill climber's process has corrupted the individual and thus reduced its fitness value, the previous genetic content of the individual is maintained and the success rate of the hill climber reduces.

The success rate of each hill climber averaged over all runs is displayed in figures 10.5 and 10.6. The success rates on each test case in figures 10.5 and 10.6 belong to first and second group hill climbers respectively.

As figure 10.5 illustrates, the hill climbers that attempt to resolve the violations arising from maximum daily workload specifications for sections perform the best for all the test cases. The ones resolving minimum workload constraints have less success rates. The hill climber that slides meetings in a section's timetable to remove gaps is the worst performer among the first group hill climbers. Among second group hill climbers, the ones that aim to resolve violations of instructor daily workload constraints again have better success rates. The hill climber that endeavors to assign at least one empty hour for travelling time between the lectures that are held at different branches for an instructor, i.e HC_I_TRAVEL, and the hill climber that tries to assure that an instructor gives lectures in at most 2 different locations, i.e branches, have 0 success rates for test cases with only one branch. Both of the hill climbers that slide meetings of sections and instructors to remove gaps have the least success rates since they are the most destructive operators. They apply the sliding process to each meeting of the course section at hand in case the assignment of the meeting causes gaps in the timetable of the current instructor or section. This would surely lead to more violations of other constraint types. Thus, the overall fitness of the current individual is reduced and the hill climber becomes unsuccessful.

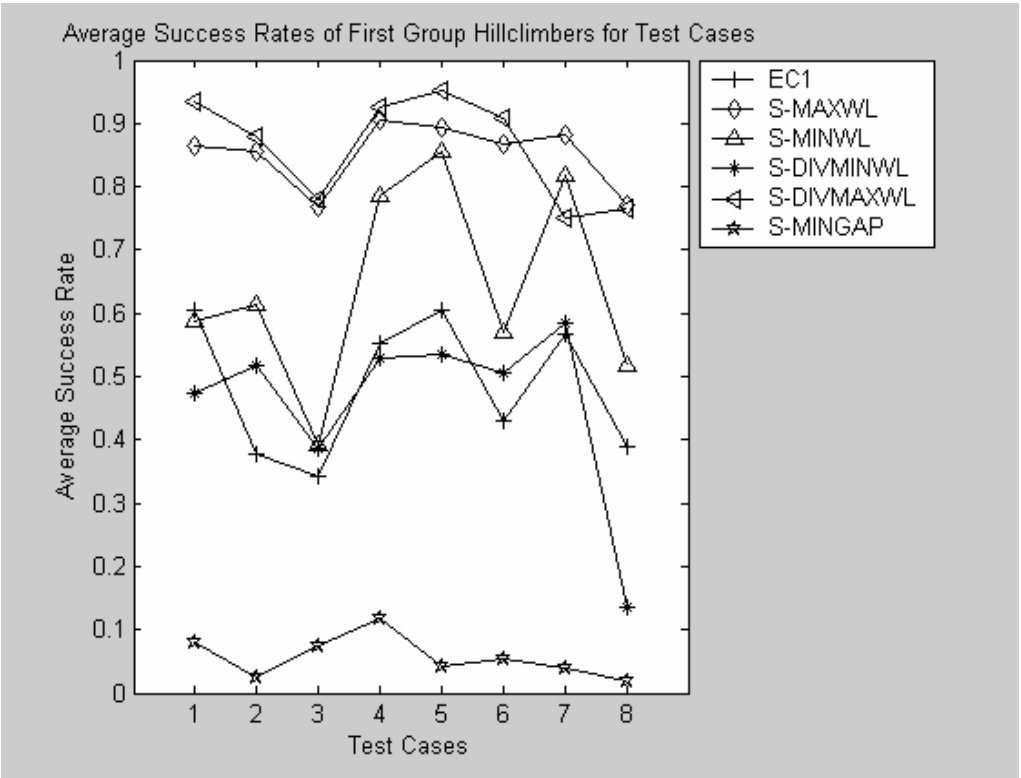

Figure 10.5 Average success rates of first group hill climbers

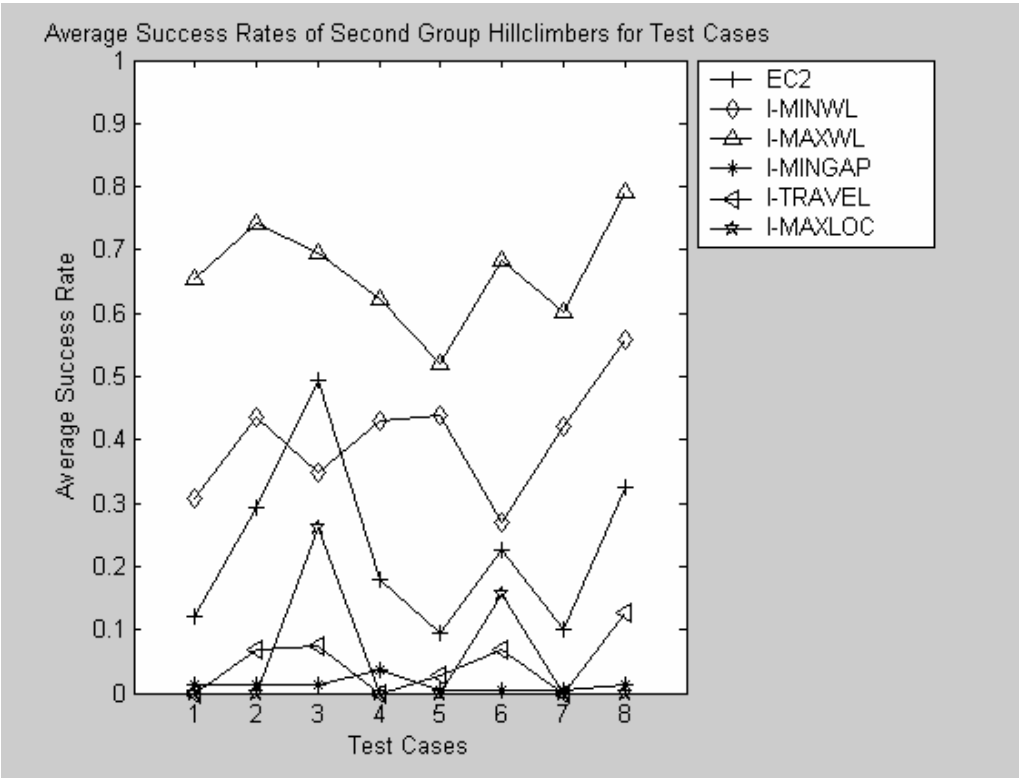

Figure 10.6 Average success rates of second group hill climbers

Hill climbers that resolve workload constraints reassign meetings to time slots where daily workload is less. They apply this procedure only to a randomly chosen meeting of the current course section. So, they are less destructive and thus better performers.

10.3.2 Results for Pure Genetic Algorithms

In the second set of experiments, the pure genetic algorithm merely employing standard genetic operators without local search is applied to the problem instances. 50 runs were carried out for each test data while allowing a maximum number of 20000 generations in each run. The population size was taken to be 100 for test cases from 1 to 6 and 200 for test 7 and 8. Since the genetic algorithm was unable to find a timetable with no hard constraint violations and no gaps in neither of the runs, each run lasted for 20000 generations. Figure 10.7 shows the average best individual fitness per run after 20000 generations for each test case. From figure 10.7, it can be argued that allowing the genetic algorithm to create more generations than 20000 would lead to better results since the results for the fitness of best individuals in each generation increases until 20000 generations. However, this many generations allowed are sufficient to compare the genetic algorithm with the proposed memetic method.

Table 10.7 Results for Pure Genetic Algorithm

	Avg. No. of Soft Constraint Violations	Avg. No. of Hard Constraint Violations	Avg. No. of Evaluations
Test 1	95.36	0.06	1,944,786.66
Test 2	199.34	0	1,944,918.54
Test 3	34.66	0	1,944,955.96
Test 4	50.80	0	1,944,935.94
Test 5	341.38	0.48	1,944,757.68
Test 6	91.70	0.06	1,944,916.10
Test 7	281.08	5.30	3,944,744.26
Test 8	275.34	0.04	3,944,591.74

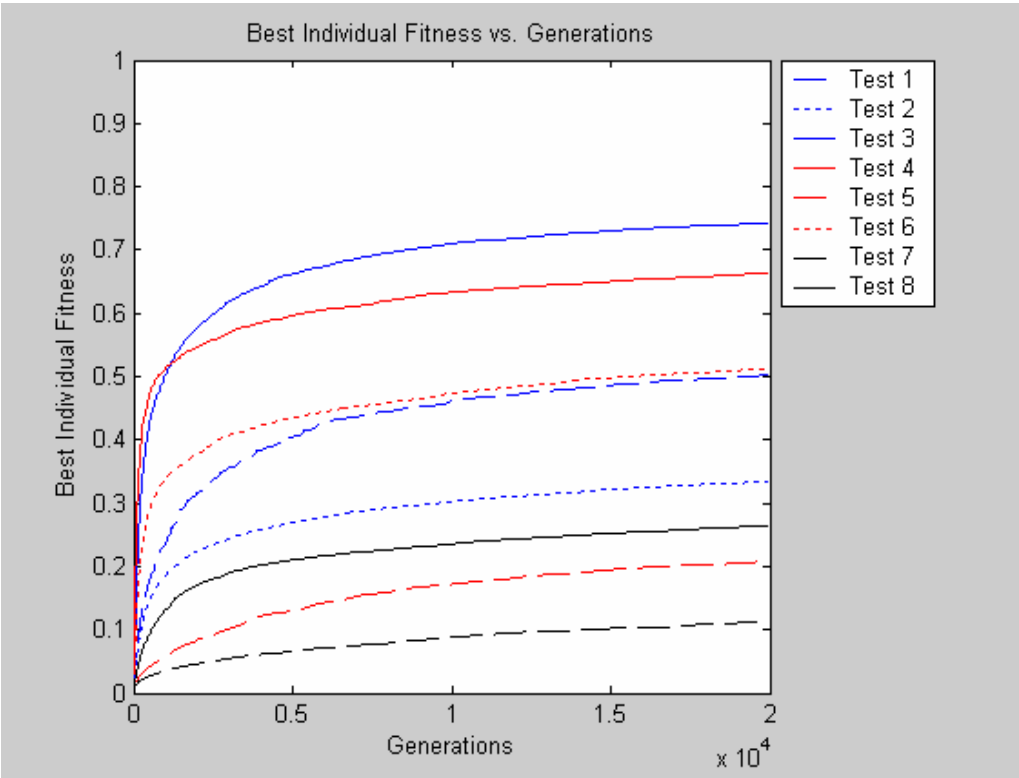

Figure 10.7 Best individual fitness in generations for pure genetic algorithm

Table 10.7 summarizes the results obtained from the second set of experiments. The first and second columns give the number of soft and hard constraint violations found in the best individual after 20000 generations. The results shown in the table are averaged over all runs. The average number of evaluations per run are also displayed in the table.

When tables 10.4 and 10.6 are compared, it is clear that pure genetic algorithm performs much poorer than the memetic algorithm with VDHC method for each test case although it is allowed to carry out much more evaluations and thus to visit much more states than the memetic algorithm. Therefore, the memetic algorithm with VDHC method obviously outperforms the pure genetic algorithm, which indicates the necessity of well managed local optimization phase.

10.3.3 Results for Multimeme Algorithms

Multimeme algorithms were experimented on 8 randomly generated test cases. The six of these problem instances have sizes varying between 108 and 378. For them, population size is 50. The other two instances have larger sizes, i.e total number of meetings, 408 and 438. For them, the population size is kept as 100. These population sizes are the same values as those kept during the experiments utilizing memetic algorithm. Each of the experiments were carried on for 50 runs and the results to be mentioned are averaged over the 50 runs.

In order to investigate the benefits of the utilization of the self-inheritance mechanism(SIM), the experiments for the multimeme algorithm are performed in two steps. In the first-step experiments, the memes for the offsprings are chosen randomly from the memes of their parents. In the second-step experiments, SIM, which requires that the offsprings acquire the memes of the fitter parent, is employed during meme crossover.

Each of the first-step multimeme experiments were carried out both with uniform and one-point crossover methods employed as global searchers. The second-step experiments which utilize SIM employed traditional uniform crossover since that recombination operator generally gives the best results.

In the first group of first and second step multimeme experiments, each individual in the population contains merely one meme. In the second group of these experiments, each individual owns memes as many as the number of different hill climbers. A meme in all of the mutimeme experiments denotes the hill climber to apply to the individual and the maximum number of unsuccessful applications of the hill climber to be allowed on the individual. All multimeme approaches employ traditional genetic operators during their evolutionary cycles.

In tables 10.8 and 10.9, UX and 1PTX refer to traditional uniform crossover and one-point crossover, respectively. Tables 10.8 and 10.9 show the average number of constraint violations along with average number of evaluations per run for the multimeme algorithm without utilizing SIM. Tables 10.10 and 10.11 show the results obtained from the multimeme algorithm with SIM. Tables 10.8 and 10.10 show results with merely one meme contained in an individual. In the experiments to obtain these results, each run continued for 5000 generations. Tables 10.9 and 10.11 show the case where an individual carries memes as many as the number of hill climbers, i.e 12 memes. In the corresponding experiments, each run continued for 500 generations. This less number of generations for this version of the multimeme algorithm is necessary since this implementation of the multimeme algorithm with memeplex size 12 performs approximately 12 times more number of evaluation and hill climbing steps than the multimeme algorithm with memeplex size equal to 1. In order to compare the algorithms better, they should be given equal chances and this is best reflected in terms of number of evaluations performed in each run.

Table 10.8 Results for multimeme algorithm with 1 meme in an individual

	Avg. Soft Constraint Violations		Avg. Hard Constraint Violations		Avg. No. of Evaluations	
	UX	1PTX	UX	1PTX	UX	1PTX
Test 1	63.62	71.56	0	0	1,974,548.76	1,975,130.44
Test 2	243.94	251.68	0	0	1,974,574.90	1,969,633.68
Test 3	51.96	53.36	0	0	1,952,355.34	1,955,005.92
Test 4	23.76	27.52	0	0	1,983,440.62	1,983,951.04
Test 5	273.56	279.94	0	0	1,973,980.68	1,976,328.20
Test 6	103.22	109.04	0	0.02	1,971,461.48	1,967,877.28
Test 7	187.18	196.32	0	0	4,040,929.48	4,031,781.16
Test 8	313.14	322.70	0	0	4,025,267.64	4,019,501.58

Table 10.9 Results for multimeme algorithm with 12 memes in individual

	Avg. Soft Constraint Violations		Avg. Hard Constraint Violations		Avg. No. of Evaluations	
	UX	1PTX	UX	1PTX	UX	1PTX
Test 1	64.88	67.80	0	0	2,083,265.88	2,072,503.00
Test 2	282.56	283.22	0	0	2,079,223.08	2,072,609.20
Test 3	86.92	91.84	0	0	2,069,506.30	2,064,981.90
Test 4	17.2	17.44	0	0	2,098,220.42	2,083,963.26
Test 5	307.30	315.48	0	0.02	2,076,575.90	2,066,322.98
Test 6	131.70	132.64	0	0	2,074,890.22	2,069,059.00
Test 7	239.32	235.08	0	0	4,268,116.20	4,218,375.50
Test 8	385.94	377.40	0	0	4,278,337.70	4,234,603.44

Table 10.10 Results for multimeme algorithm utilizing SIM with 1 meme in individual

	Avg. Soft Constraint Violations	Avg. Hard Constraint Violations	Avg. No. of Evaluations
Test 1	68.26	0	1,962,226.12
Test 2	243.10	0	1,968,741.84
Test 3	50.30	0	1,954,309.56
Test 4	22.36	0	1,987,404.34
Test 5	275.38	0	1,971,033.34
Test 6	105.54	0.02	1,972,612.30
Test 7	187.78	0	4,035,978.30
Test 8	312.34	0	4,010,079.84

Table 10.11 Results for multimeme algorithm utilizing SIM with 12 memes in individual

	Avg. Soft Constraint Violations	Avg. Hard Constraint Violations	Avg. No. of Evaluations
Test 1	68.54	0	2,085,621.96
Test 2	282.48	0	2,082,281.04
Test 3	89.86	0	2,072,443.30
Test 4	17.52	0	2,095,085.42
Test 5	314.00	0	2,074,950.94
Test 6	132.10	0	2,075,960.52
Test 7	239.06	0	4,272,457.92
Test 8	378.70	0	4,280,366.48

This conclusion can also be drawn from the fact that evaluation of individuals is the bottleneck of genetic algorithms.

The first thing to notice from the tables belonging to the multimeme algorithm is that all versions of the multimeme algorithm implemented have outcomes worse than those of the memetic algorithm for all the test cases.

Secondly, the utilization of SIM for the multimeme algorithm does not enhance the overall performance in most of the cases. The fitness of best individuals is even slightly degraded while employing SIM. Figure 10.8 provides means for better comprehension of the meme concentration in each generation for random and SIM versions of the multimeme algorithm with one meme in an individual. The meme configurations with top 5 average concentrations are displayed. All these memes denote the hill climber HC_EC1 and they have values from 5 to 9 for the maximum number of unsuccessful iterations. Blue lines refer to the results when SIM is utilized, while red lines show the results for random inheritance of memes in the population. The figure reveals that there is little difference between the concentrations belonging to the most intensive 5 memes for the two versions of the algorithm. The approximately equal meme concentration gives an explanation for the approximately equal results obtained while utilizing SIM or random meme inheritance.

Figure 10.9 shows the best individual fitness vs. generations both for SIM utilization and random inheritance on test case 1. The slight degregation of results brought about by SIM is clearly observed from the figure. Therefore, for private school timetabling problem, inheriting the memes of the fitter parent to offsprings does not improve the performance of the algorithm. The result can be traced to the fact that memes of the parents resemble each other as time passes.

Figure 10.10 compares the results of the multimeme experiments with 1 and 12 memes for an individual on test 8 problem instance. It displays

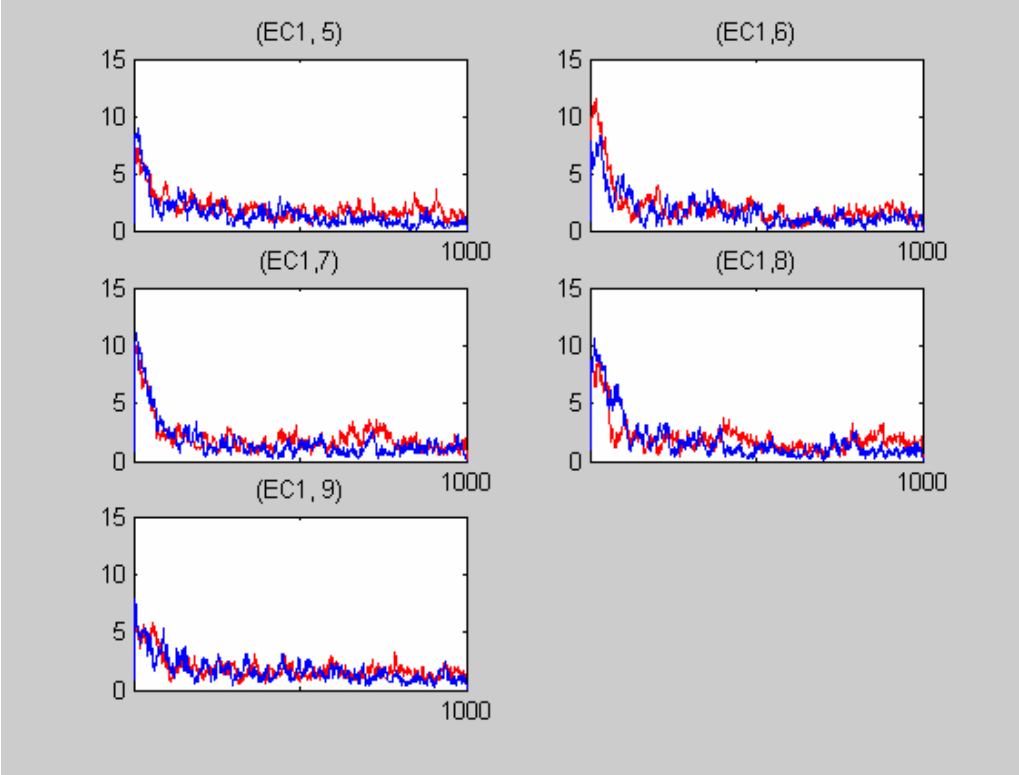

Figure 10.8 Meme concentration in first 1000 generations for multimeme algorithm with 1 meme on test case 1. Memes with top 5 average concentration are displayed. Blue lines indicate utilization of SIM. Red lines denote the results for random meme inheritance.

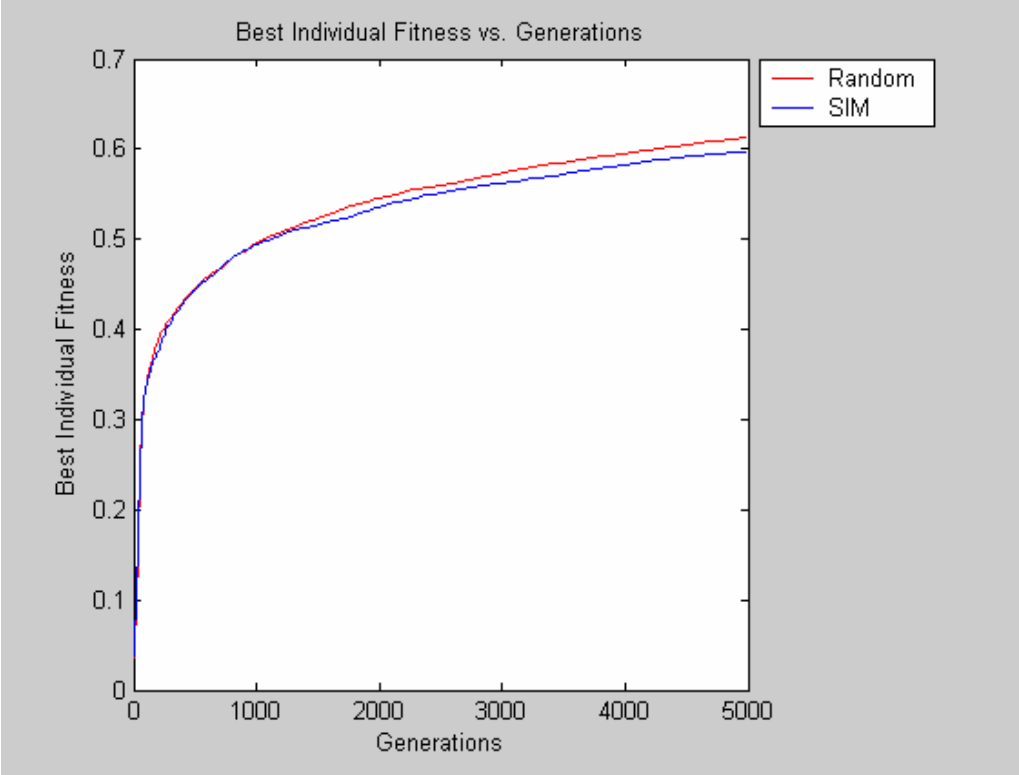

Figure 10.9 Best individual fitness in generations for multimeme algorithm with 1 meme in individual for test case 1.

the fitness of the best individual in each generation for the two algorithms.

When we try to compare best fitness values in terms of number of generations passed, the multimeme algorithm where an individual has 12 memes seems to be doing better. However, this way of comparison leads to a wrong interpretation of results. Hence, Figure 10.10 is obviously misleading and has been intentionally put in this study to point out the mistake of comparing two such algorithms in terms of generations.

In a generation of an evolutionary algorithm with local search, each individual enters through a local optimization phase. In the multimeme algorithm with memeplex size equal to 1, each individual is applied only one hill climbing operator. However, in the other version of the algorithm where each individual has 12 memes, each newly created individual is applied possibly different 12 hill climbing operators. If we keep the maximum number of unsuccessful applications of a hill climber in a meme constant, it is clear that there will be much more individual evaluations during a generation of the multimeme algorithm with memeplex size equal to 12 than the multimeme algorithm with memeplex size equal to 1. Here, it is assumed that the memeplex is composed of the memes that the individual carry and thus its size refers to the number of memes belonging to each individual. Therefore, the average number of evaluations per run should be consulted for the real performance of the two versions of multimeme algorithms.

By examining the tables for the results of the multimeme algorithm, the following observation can be made: In the results for most of the test instances, multimeme algorithm with memeplex size equal to 12 found best timetables with more number of soft constraint violations although it performed more number of evaluations than the multimeme algorithm with memeplex size equal to 1. Therefore, increasing the number of memes in an individual cannot lead to a performance increase in the private school timetabling.

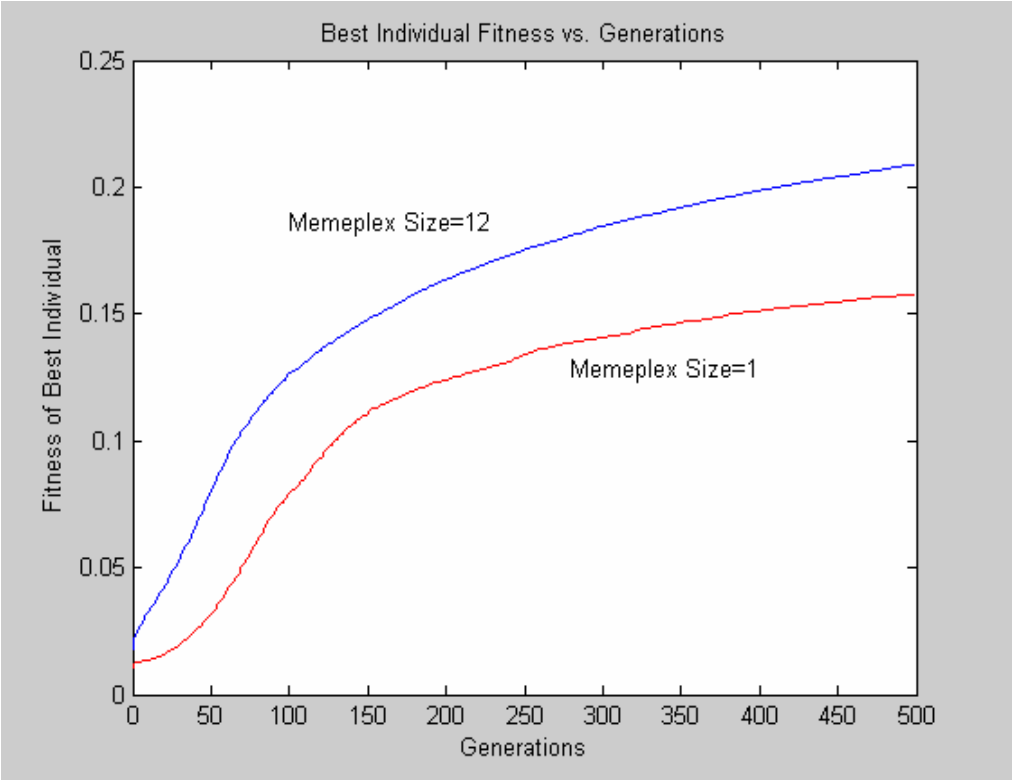

Figure 10.10 Best fitness vs. generations for multimeme algorithms on test 8 data.

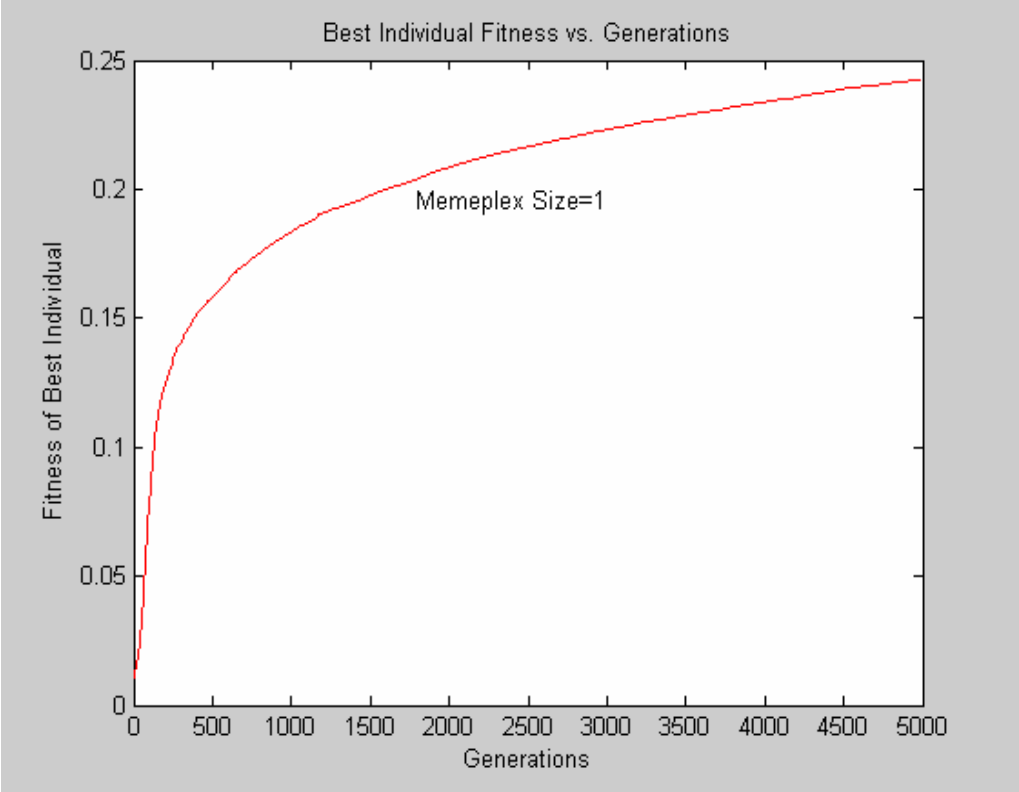

Figure 10.11 Best fitness vs. generations for multimeme algorithm with memeplex size 1 on test 8 data.

Figure 10.11 also contributes to the indication of the success of multimeme algorithm with only one meme in each individual over the other multimeme algorithm with memeplex size equal to 12 on test 8 problem instance. The fitness value of the best individual that the first algorithm reaches after 5000 generations is better than that of the latter after 500 generations. However, although the first one went through 5000 generations, the number of evaluations it performed during a run on the average is 4,010,079.84, which is less than the number of evaluations the latter performed, namely 4,280,366.48.

CHAPTER 11

CONCLUSION

In this study, a new timetabling problem for private schools in Turkey have been discussed. A memetic algorithm employing VDHC method has been designed to solve the synthetic instances of this problem. The experiments for the randomly generated private school timetabling problem instances have been attempted with pure genetic algorithms and multimeme algorithms as well as the proposed memetic algorithm.

Among the new global search operators proposed, violation-directed mutations gave slightly better results than the traditional mutation operator. However, uniform crossover gave the best results among all the newly proposed crossover operators.

The results obtained clearly reveal that the utilization of VDHC method for memetic algorithms greatly enhances the results for all the test cases. This can be traced to the fact that the success rates of the hill climbers are most increased with the application of the VDHC method. The VDHC method provides hill climbers with effective management and coordination. It chooses the hill climbers whose corresponding violations are more concentrated on the chromosome and guides them to the areas of the chromosome where there are more violations for them to resolve.

Although the pure genetic algorithm is allowed to evaluate more states in the search space than those for the memetic algorithm per run, it turns out to give worse results for all the test cases.

The results for multimeme algorithms are even worse than those of the memetic algorithm where hill climbers to act on the individual are randomly chosen without the utilization of VDHC method.

As a result, memetic algorithms come out as the best choice for the private school timetabling problem, especially when their low-level operators are managed by efficient hyper-heuristics.

REFERENCES

Abramson, D. (1991a) "Constructing School Timetables Using Simulated Annealing", Sequential and Parallel Algorithms, Management Science, vol. 37, pp. 98-113.

Abramson, D. and Abela, J. (1991b) "A Parallel Genetic Algorithm for Solving the School Timetabling Problem", Technical Report, Division of Information Technology, C.S.I.R.O.

Alkan, A. and Ozcan, E. (2003) "Memetic Algorithms for Timetabling", Proc. of 2003 IEEE Congress on Evolutionary Computation, pp. 1796-1802.

Bufe', M., Fischer, T., Gubbels, H., Hacker, C., Hasprich, O., Scheibel, C., Weicker, K., Weicker, N., Wenig, M. and Wolfangel, C. (2001) "Automated Solution of a Highly Constrained School Timetabling Problem - Preliminary Results", Proc. of the EvoWorkshops on Applications of Evolutionary Computing, pp. 431 – 440.

Burke, E.K., Newall, J.P. and Weare, R.F. (1995a) "A Memetic Algorithm for University Exam Timetabling", Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, pp. 241 – 250.

Burke, E.K., Elliman, D., Ford, P. H., and Weare, R. F. (1995b) "Examination Timetabling in British Universities: A Survey", PATAT 1995: 76-90E.

Burke, E. K., Elliman, D.G. and Weare, R.F. (1995c) "The Automation of the Timetabling Process in Higher Education", Journal of Educational Technology Systems, vol. 23, no 4, pp. 257-266, Baywood Publishing Company.

Burke, E.K., Cowling, P. and Landa Silva, J.D. (2001a) "Hybrid Population-based Metaheuristic Approaches for the Space Allocation Problem", Proc. of the 2001 Congress on Evolutionary Computation (CEC 2001), IEEE press, pp. 232-239.

Burke, E.K., Cowling, P., De Causmaecker, P. and Vanden Berghe, G. (2001b) "A Memetic Approach to the Nurse Rostering Problem", Applied Intelligence, 15(3), pp. 199-214.

Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P. and Schulenburg, S. (2003) "Hyper-heuristics: An Emerging Direction In Modern Search Technology", Ch. 16 in Handbook of Meta-Heuristics, pp. 457–474, Kluwer Academic Publishers.

Burke, E.K. and Landa Silva, J.D. (2004) "The Design of Memetic Algorithms for Scheduling and Timetabling Problems", Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing, vol. 166, Springer, pp. 289-312.

Collingwood, E., Ross, P. and Corne, D. (1997) "A Guide to GATT".

Colorni, A., Dorigo, M. and Maniezzo, V. (1990) "A Genetic Algorithm to Solve the Timetable Problem", Technical Report 90-060, Politecnico di Milano.

Corne, D. and Ross, P. http://www.dcs.napier.ac.uk/~peter/sw/clump.c, Last Access Date: 06/21/2005.

Corne, D., Ross, P. and Fang, H. (1994) "Fast Practical Evolutionary Timetabling", Proc. of AISB Workshop on Evolutionary Computing.

Erben, W. (1995) "A Grouping Genetic Algorithm for Graph Coloring and Exam Timetabling", The Practice and Theory of Automated Timetabling III: Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling (PATAT 1995), Lecture Notes in Computer Science, 1153, pp. 198-211, Springer.

Even, S., Itai, A. and Shamir, A. (1976) "On the Complexity of Timetable and Multicommodity Flow Problems", SIAM J. Comput., 5(4): pp. 691-703.

Fang, H. (1994) "Genetic Algorithms in Timetabling and Scheduling", Ph.D. Thesis, Department of Artificial Intelligence, University of Edinburgh, Scotland.

Filho, G. R. and Lorena, L. A. N. (2001) "A Constructive Evolutionary Approach to School Timetabling" , Lecture Notes In Computer Science, vol. 2037, Proc. of the EvoWorkshops on Applications of Evolutionary Computing, pp. 130 – 139.

Furtado, J. C. and Lorena, L. A. N. (1998) "Constructive Genetic Algorithms for Clustering Problems", Evolutionary Computation.

Junginger, W. (1986) "Timetabling in Germany – a Survey", Interfaces, vol. 16, pp. 66-74.

Hentenryck, P. V. and Saraswat, V. A. (1997) "Constraint Programming: Strategic Directions", Constraints Journal, vol. 2, pp. 7–33.

Holland, J.H. (1975) "Adaptation in Natural and Artificial Systems", The University of Michigan Press.

Krasnogor, N., Smith, J.E. (2001) "Emergence of Profitable Search Strategies Based on a Simple Inheritance Mechanism", Proc. of the 2001 Genetic and Evolutionary Computation Conference, Morgan Kaufmann.

Krasnogor., N. (2002a) "Memetic Algorithms", A Tutorial given in the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII), Granada, Spain.

Krasnogor, N. (2002b) "Studies on the Theory and Design Space of Memetic Algorithms", PhD Thesis, University of the West of England, Bristol, United Kingdom.

Krasnogor, N. and Gustafson, S. (2003a) "The Local Searcher as a Supplier of Building Blocks in Self-Generating Memetic Algorithms", Fourth International Workshop on Memetic Algorithms (WOMA IV) Workshop, Proc. of the 2003 Genetic and Evolutionary Computation Conference, GECCO 2003, Chicago, USA.

Krasnogor, N. and Gustafson, S. (2003b) "A study on the use of "Self-Generation" in Memetic Algorithms".

Marte., M. (2000) "Towards Constraint-based Grammar School Timetabling", Proc. of the 3rd International Conference on the Practice and Theory of Automated Timetabling, pp. 222–224.

Marte, M. (2003) "Models and Algorithms for School Timetabling – A Constraint-Programming Approach", Doctoral thesis, Institut für Informatik der Universität München.

Miranda, V. and Proença, L. M. (1999) "Genetic/Evolutionary Algorithms and Application to Power Systems", ISAP'99 - Intelligent Systems Application to Power Systems, Tutorial Course Book.

Moscato, P. and Norman, M.G. (1992) "A `Memetic' Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems", Parallel Computing and Transputer Applications, Amsterdam, pp. 187-194.

Müller, T. (2002) "Some Novel Approaches to Lecture Timetabling", CPDC'.

Terashima-Marin, H., Ross, P.M. and Valenzuela-Rendon, M. (1999) "Evolution of Constraint Satisfaction Strategies in Examination Timetabling", Proc. of the GECCO-99 Genetic and Evolutionary Computation Conference, pp 635-642. Morgan Kaufmann.

OSYM, Selection and Placement of Students in Higher Education Institutions in Turkey, A condensed English Version. http://www.osym.gov.tr/BelgeGoster.aspx?DIL=1&BELGEBAGLANTIANAH=169, Last Access Date: 06/21/2005.

Ozcan, E. and Ersoy, E. (2005a) "Final Examination Scheduler – FES", IEEE Congress on Evolutionary Computation, to appear.

Ozcan, E. and Ersoy, E. (2005b) "Solving Timetabling Problems Using Memetic Algorithms", in review.

Radcliffe, N.J. and Surry, P.D. (1994) "Formal Memetic Algorithms", Evolutionary Computing: AISB Workshop.

Ross, P., Corne, D. and Fang, H. L. (1994a) "Improving Evolutionary Timetabling with Delta Evaluation and Directed Mutation", Parallel Problem Solving from Nature III, Springer Verlag.

Ross, P., Corne, D. and Fang, H. L. (1994b) "Successful Lecture Timetabling with Evolutionary Algorithms", Proc. of the ECAI 94 Workshop on Applications of Evolutionary Algorithms.

Santos, H.G., Ochi, L.S and Souza, M. J. F. (2004) "An Efficient Tabu Search Heuristic for the School Timetabling Problem", Lecture Notes in Computer Science, vol. 3059, , p. 468-481.

Schaerf, A. and M. Schaerf. (1995) "Local Search Techniques for High-school timetabling", Proc. of the 1st Intl. Conf. On the Practice an Theory of Automated Timetabling, pp. 313-323.

Schaerf, A. (1996) "Tabu Search Techniques for Large High-School Timetabling Problems", Proc. of the Thirteenth National Conference on Artificial Intelligence (AAAI'96), AAAI Press/MIT Press.

Schaerf, A. (1999) "Local Search Techniques for Large High-school Timetabling Problems", IEEE Transactions on Systems, Man, and Cybernetics, 29(4), 368-377.

Schaerf, A. and Di Gaspero, L. (2001) "Local Search Techniques for Educational Timetabling Problems", Proc. of the 6th International Symposium on Operational Research in Slovenia (SOR-'01), Preddvor, Slovenia, pp.13-23.

Schmidt, G. and Strohlein, T. (1979) "Timetable construction – an annotated bibliography", The computer Journal, 23(4), 307-316.

Souza, M. J. F., Ochi, L. S. and Maculan, N. (2003) "A GRASP-Tabu Search Algorithm for Solving School Timetabling Problems", Metaheuristics: Computer Decision-Making, Kluwer Academic Publishers, pp. 659-672.

Landa Silva, J.D (2003) "Metaheuristic and Multiobjective Approaches for Space Allocation", Doctoral Thesis, School of Computer Science and Information Technology, University of Notthingham.

Welsh, D.J.A. and M.B. Powell, M.B. (1967) "An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems", Comp. Jrnl. 10, pp. 85-86.

Whitley., D. (1989) "The GENITOR Algorithm and Selection Pressure", Proc. of the Third International Conference on Genetic Algorithms, pp. 116-121.

APPENDIX A.

A.1 Parameters for RDG

Table A.1 Parameters of Problem Instance Size for RDG

Parameter Name	Description
min_grade_no	minimum number of available grades in a branch
max_grade_no	maximum number of available grades in a branch
min_div_no	minimum number of available divisions in a grade
max_div_no	maximum number of available divisions in a grade
min_course_no	minimum number of available courses for students in a division
max_course_no	maximum number of available courses for students in a division
min_meeting_no	minimum number of available meetings in a course
max_meeting_no	maximum number of available meetings in a course
min_meeting_length	minimum available meeting length in a course
max_meeting_length	maximum available meeting length in a course
min_branch_no	minimum number of branches in a problem instance
max_branch_no	maximum number of branches in a problem instance
min_section_no	minimum number of sections in a division
max_section_no	maximum number of sections in a division

Table A.2 Curricular Parameters for RDG

Parameter Name	Description
min_own_div_course_no	minimum number of courses taken from its own division for a section
max_own_div_course_no	maximum number of courses taken from its own division for a section
min_other_divs_no	minimum number of other divisions that a section takes course from
max_other_divs_no	maximum number of other divisions that a section takes course from
min_other_div_course_no	minimum number of courses taken from a division other than its own for a section
max_other_div_course_no	maximum number of courses taken from a division other than its own for a section
min_offered_courses_no	minimum number of courses offered from a division
max_offered_courses_no	maximum number of courses offered from a division

Table A.3 Instructor Parameters for RDG

Parameter Name	Description
min_assigned_course_no	minimum number of different courses an instructor can teach
max_assigned_course_no	maximum number of different courses an instructor can teach

Table A.4 Temporal Parameters for RDG

Parameter Name	Description
min_daily_hour_no	minimum number of daily available hours in a problem instance
max_daily_hour_no	maximum number of daily available hours in a problem instance
min_slot_no	minimum number of total slots for a term
max_slot_no	maximum number of total slots for a term
min_daily_assigned_slot_no	minimum number of assigned slots for a grade in a day
max_daily_assigned_slot_no	maximum number of assigned slots for a grade in a day
min_begin_hour	minimum daily beginning hour for assigned meetings of a section
max_begin_hour	minimum daily beginning hour for assigned meetings of a section
min_grade_day_no	minimum number of days available for a grade
max_grade_day_no	maximum number of days available for a grade
min_day_no	minimum number of defined days for a problem instance
max_day_no	maximum number of defined days for a problem instance

Table A.5 Parameters of Constraint Density for RDG

Parameter Name	Description
instructor_exclusion_probability	probability that determines whether an instructor defines exclusion slots
instructor_slot_exclusion_probability	probability that determines which slots an instructor will exclude
instructor_workload_constraint_probability	probability that determines whether an instructor defines workload constraints
course_section_preset_probability	probability that determines whether a course section defines preset time slots
section_exclusion_probability	probability that determines whether a section defines exclusion slots
section_daily_exclusion_probability	probability that determines which days of the section will have excluded slots
section_hourly_exclusion_probability	probability that determines which hours of the section will be excluded in a chosen day
section_daily_workload_constraint_probability	probability that determines whether a section defines workload constraints

A.2 Output for the RDG

The output file will begin with the initial information in the format displayed in Table A.6.

Table A.6 Representation of Initial Points in the Problem Instance

```
There is(are) 2 branch(es)
In Branch 0
        There is(are) 4 grade(s)
        In Grade 0
                There is(are) 4 division(s)
                In Division 0
                        There is(are) 4 section(s)
                        Section(s) is(are):
                                S 0 S 1 S 2 S 3
                                S 0 is assigned to 4 course sections
                                        These course sections are CS 0 CS 1 CS 2 CS
        3
                                ...
                In Division 1
                                ...
        In Grade 1
                ...
```

As seen above, the information about the size of the problem will be listed in a hierarchical way. Every branch, grade and division has an identification number that is unique among the set it is contained. To ilustrate, every branch has a unique number. Every grade in a branch have identification numbers that are unique among the set of grades for the current branch. Similarly, the identification numbers of every division in a specific grade at a particular branch, i.e each division of grade 1 at branch 0, differ from each other. However, sections have identification values that are unique among the set of all the sections belonging to the problem instance. The same property applies for the course sections as well.

The curricular information is outputted as displayed in Table A.7.

Table A.7 Representation of Curricular Information in the Problem Instance

```
There is(are) 5 possible grade(s)
In Grade 0
        There is(are) 4 possible-division(s)
        In Division 0
                There is(are) 4 course(s)
                In Course 6 offered by division 0
                        There is(are) 4 meeting(s)
                        Meeting_Length(s) is(are): 2 2 1 1
                In Course 1 offered by division 0
                        There is(are) 2 meeting(s)
                        Meeting_Length(s) is(are): 2 1
                In Course 3 offered by division 0
                        There is(are) 3 meeting(s)
                        Meeting_Length(s) is(are): 2 1 1
                In Course 4 offered by division 0
                        There is(are) 3 meeting(s)
                        Meeting_Length(s) is(are): 1 1 1
        In Division 1
        ...
In  Grade 1
        ...
```

The curricular information defines the possible grades to be offered and the possible divisions in each of the grades. The courses listed for a division show the curriculum of sections that belong the current grade and division.

Each division offers a number of courses to the sections that are from different divisions as well as the ones that belong to it. Each course has a unique identification within the set of courses that are also offered from its division. For instance, course 6 offered from division 0 can be interpreted as "Physics course from quantitative division". It is maintained via parameters that the number of courses offered from its own division are more in number, have more meetings and/or have

longer meetings in the curriculum for a section. To illustrate, a section from verbal division have more courses offered from the verbal division such as geography, etc. The number of days and daily hours defined are outputted as in Table A.8.

Table A.8 Representation of Temporal Structure in the Problem Instance

```
There is(are) 7 day(s)
There is(are) 11 daily_hour(s)
In Branch 0
Grade 0 has 53 slots
1 2 3 4 5 6 7 8 11 12 13 14 15 16 17 18 24 25 26 27 28 29 30 31 32 45 46
47 48 49 50 51 52 53 54 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
72 73
Grade 1 has 55 slots
2 3 4 5 6 7 8 9 10 13 14 15 16 17 18 19 20 21 25 26 27 28 29 30 31 32 35
36 37 38 39 40 41 42 43 45 46 47 48 49 50 51 52 53 54 56 57 58 59 60 61
62 63 64 65
...
```

Then, the available time slots for all the grades in each of the branches are listed. The time slots are numbered from 0 to (number of days x daily hours). It is revealed via interviews with private school authorities that all the grades in a particular branch have a predefined set of time slots and each section belonging to that grade is assigned to course section meetings from that set. Instructor assignments are listed for all the instructors as in Table A.9

Table A.9 Representation of Instructor Assignments in the Problem Instance

```
Instructor 0 from division 0 teaches 3 course section(s)
He(She) teaches course sections CS 0 CS 5 CS 456
```

Every instructor belongs to a specific division and can lecture only courses from that division. There is a maximum limit defined on the number of different courses an instructor can teach.

Preset and exclusion constraints are the constraints that indicate the specified time slots for course sections or excluded time slots for sections or instructors. Knowing that each slot in the timetable is enumerated, each of those constraints are listed for the sections, course sections and instructors that have defined such constraints. Those sections, course sections and instructors are probabilistically chosen by the aid of parameters. Below are some portions of the output file that display a few of the preset and exclusion constraints.

Table A.10 Representation of Exclusions and Specifications in the Problem Instance

```
...
Meetings of Course Section 53 should be held at slots: 46 57 67 4 13
...
Section S 0 excludes 8 slots
These slots are 15 16 17 18 26 28 29 30
...
Instructor 6 excludes 11 slots
These slots are  6 8 11 31 34 51 55 61 63 73 75
Instructor 7 excludes 0 slots
...
```

Binary event constraints, i.e the edge constraints are not reproduced in the output file since they are present in the definition of the problem. The event spread constraints below are listed in the output file for each of the instructors or sections that have them. Those sections and instructors are probabilistically chosen by the aid of parameters.

Table A.11 Representation of Event-spread Constraints in the Problem Instance

```
...
Instructor 56 should lecture minimum 1 maximum 4 hours a day
...
Instructor 172 should lecture minimum 2 maximum 3 hours a day
...
In a day, Section S 64 can be assigned Maximum 7 minimum 5 hours totally
In a day, this section can be assigned to courses from at most 2 divisions
There should be
        Maximum 5 minimum 3 hours for courses offered by division 1
        Maximum 3 minimum 1 hours for courses offered by division 2
...
```

The instructor constraints other than the instructor exclusion constraints and workload constraints are not reproduced in the output file since they are present in the definition of the problem.

The sample file in Table A.12 returns the analysis of a specific problem instance created by the RDG.

Table A.12 Analysis of the Randomly Generated Data

There are 8 days
There are 10 daily hours
There are 960 course section meetings
There are 384 course sections
There are 108 sections
There are 27 divisions
There are 9 grades
There are 3 branches

Conflict Density between course section meetings is 0.043904
Conflict Count(Edge Constraints only EC1 3084, only EC2 16358, both 768) between course section meetings is 20210
Percent of course sections that have preset slots for their meetings is 6.250000
Percent of meetings whose time slots have been predefined is 6.041667

Total number of Sections is 108
Average number of course sections for a section is 3.555556
Average number of meetings for a section is 8.888889
Average number of hours for a section is 10.444444
Max. number of meetings for a section is 11
Max. number of hours for a section is 12
Average number of available slots for a section is 79.814815
Percent of Sections who define slot exclusions is 100.000000
Average number of excluded slots for those sections is 0.185185
Percent of Sections who define workload constraints is 48.148148
Percent of Sections who define workload constraints for courses from each offering division is 48.148148

Total number of Instructors is 29
Average number of meetings for an instructor is 33.103448
Average number of hours for an instructor is 38.896552
Max. number of meetings for an instructor is 53
Max. number of hours for an instructor is 59
Average number of available slots for an instructor is 79.241379
Percent of Instructors who define slot exclusions is 6.896552
Average number of excluded slots for those instructors is 11.000000
Percent of Instructors who define workload constraints is 55.172414