

UNIVERSITY COURSE TIMETABLING USING MULTI OBJECTIVE GENETIC  
ALGORITHMS

by  
Ahmet Ulak


Submitted to the Institute of Graduate Studies in  
Science and Engineering in partial fulfillment of  
the requirements for the degree of  
Master of Science  
in  
Computer Engineering

Yeditepe University  
2010

UNIVERSITY COURSE TIMETABLING USING MULTI OBJECTIVE GENETIC  
ALGORITHMS

APPROVED BY:

Assoc. Prof. Dr. Emin Erkan Korkmaz  
(Supervisor)




.....

Assist. Prof. Dr. A. Şima Etaner Uyar



.....

Assist. Prof. Dr. Onur Demir



.....

DATE OF APPROVAL: 14/12/2010

*To my mother, father and my nephews Zeynep, Arda and Ege..*

## ACKNOWLEDGEMENTS

I sincerely thank Asoc. Prof. Dr. Emin Erkan Korkmaz for his excellent guidance. His valuable advises and corrections made this thesis become real.

I also sincerely thank Assist. Prof. Dr. A. Şima Etaner Uyar and Assist. Prof. Dr. Onur Demir for their excellent guidance. Again their valuable advises and corrections made this thesis become real.

I am grateful to Asoc. Prof. Dr. Ender Özcan. Without his guidance I wouldn't start my academic career.

I sincerely thank Prof. Dr. Şebnem Baydere for her encouragement.

I would like to thank Assist. Prof. Dr. Dionysis Goularas, Assist. Prof. Dr. Gürhan Küçük, Dr. Mustafa Mutluoğlu and Mr. Mehmet Aksayan for motivating and assisting me whenever I needed.

I would like to thank other CSE members also since they gave me much.

I also thank to my colleagues Ahmet Atasoy, Kerem Irgan, Nazlı Nakeeb, Buse Yılmaz, Berna Kiraz and Fatih Uzun for their assistance.

I am also grateful to my parents, friends and a special friend for their patience. Without my mother's cooking and meals I would starve and this thesis couldn't become real.

Lastly I am very grateful to my father who always supported me for studying.

## **ABSTRACT**

### **UNIVERSITY COURSE TIMETABLING USING MULTI OBJECTIVE GENETIC ALGORITHMS**

University Course Timetabling is one of the interesting and exciting research areas of combinatorial optimization. This is not only due to the NP-hardness of the problem but also diversity of the researches in this area. University course timetables should be feasible and of course decent. The timetables created should satisfy different constraints which can be classified as hard and soft. Hard constraints should be obeyed strictly so that the timetable becomes feasible. On the other hand, a decent timetable should also satisfy the soft constraints. Experience in this area shows that these constraints conflict with each other in most of the cases. Using graph coloring algorithms is an approach to satisfy hard constraints. In this work, multi objective genetic algorithms are used to solve Yeditepe University Computer Engineering Department's course timetabling problem. Some special constraints of Yeditepe University Computer Engineering Department which are not common in timetabling literature are handled. The genetic algorithm utilized handles the hard constraints as a graph coloring problem and solves these constraints together with the soft ones using the multi objective framework. The adopted version of this framework is also applied to solve the benchmarking timetabling problems proposed by University of Udine - Italy.

## ÖZET

### ÇOK HEDEFLİ GENETİK ALGORİTMALAR KULLANARAK ÜNİVERSİTE DERS PROGRAMLAMASI

Yapay Zeka alanındaki en ilgi çekici ve heyecan verici konulardan birisi Üniversite Ders Programlamasıdır. Bu sadece problemin çok zor olmasından değil aynı zamanda bu konuda çok çeşitli araştırma yapılmış olmasından da kaynaklanır. Üniversite Ders Programları işlevsel olmalarının yanında kaliteli de olmalıdır. Kaliteli bir ders programı zorunlu olmayan kısıtları da sağlamalıdır. Genellikle bu kısıtların bir çoğu birbiriyle çelişmektedir. Çizge Boyama (GC) algoritması zorunlu kısıtları çözümlenmede kullanılan yaklaşımlardan birisidir. Bu çalışmada çok hedefli genetik algoritmalar Yeditepe Üniversitesi Ders Programı problemini çözümlenmede kullanılmıştır. Yeditepe Üniversitesi Bilgisayar Mühendisliği bölümünün kendisine özgü birtakım kısıtlar çözümlenmeye çalışılmıştır. Aynı yöntem İtalya'daki Udine Üniversitesi'nin ölçüm karşılaştırma testlerinde de kullanılmıştır.

## TABLE OF CONTENTS

DEDICATION.....	iii
ACKNOWLEDGEMENTS.....	iv
ABSTRACT .....	v
ÖZET .....	vi
TABLE OF CONTENTS .....	vii
LIST OF FIGURES .....	viii
LIST OF TABLES.....	ix
LIST OF SYMBOLS/ABBREVIATIONS.....	x
1. INTRODUCTION .....	1
2. TIMETABLING .....	5
2.1. RELATED WORK.....	5
3. GRAPH COLORING .....	13
3.1. RECENT APPROACHES ON REPRESENTATION .....	13
4. GENETIC ALGORITHMS .....	17
4.1. GENETIC ALGORITHMS BASICS .....	18
4.1.1. Selection .....	18
4.1.2. Crossover .....	19
4.1.3. Mutation.....	20
4.2. MULTIOBJECTIVE GENETIC ALGORITHMS .....	20
4.3. PARETO OPTIMIZATION .....	28
5. MOGA FRAMEWORK .....	29
5.1. UNIVERSITY COURSE TIMETABLING .....	29
5.1.1. Yeditepe University CSE Department's Time Table .....	30
5.1.2. University of Udine Benchmarks and ITC2007 .....	32
5.2. MOGA DESIGN .....	33
5.2.1. Chromosome representation and encoding.....	33
5.2.2. Population .....	36
5.2.3. Fitness function.....	37
5.3. FLOW OF ALGORITHM.....	40

5.3.1. Initialization and Evaluation.....	41
5.3.2. Selection .....	42
5.3.3. Crossover .....	44
5.3.4. Mutation.....	45
5.3.5. Hillclimbing.....	46
5.3.6. Next Generation.....	46
6. TESTS.....	49
6.1. YEDİTEPE UNIVERSITY CSE UCT TESTS .....	49
6.2. BENCHMARK TESTS .....	54
7. CONCLUSION.....	62
8. REFERENCES .....	63



## LIST OF FIGURES

Figure 2.1. Sector based crossover.....	11
Figure 3.1. LLE and its corresponding array representation.....	14
Figure 4.1. Structure of DNA and a gene.....	17
Figure 4.2. Pareto ranking.....	21
Figure 4.3. Sharing distance.....	22
Figure 4.4. Crowding distance .....	26
Figure 5.1. Group number encoding scheme .....	33
Figure 5.2. Flowchart .....	41
Figure 5.3. MOGA example.....	48
Figure 6.1. Comparison of population sizes for hard conflicts .....	50
Figure 6.2. Comparison of population sizes for soft conflicts .....	50
Figure 6.3. Single vs. Multiobjective .....	52
Figure 6.4. The effect of applying hill climbers.....	55
Figure 6.5. The effect of applying hill climbers.....	55
Figure 6.6. Comparison of selection crossover pairs for instance comp18.ctt .....	57

Figure 6.7. Comparison of selection crossover pairs for instance comp18.ctt .....	58
Figure 6.8. Comparison of selection crossover pairs for instance comp03.ctt .....	58
Figure 6.9. Comparison of selection crossover pairs for instance comp03.ctt .....	59
Figure 6.10. Comparison of selection crossover pairs for instance comp09.ctt .....	59
Figure 6.11. Comparison of selection crossover pairs for instance comp09.ctt .....	60

## LIST OF TABLES

Table 4.1. Domination example.....	25
Table 5.1. Group number to timeslots mapping .....	34
Table 5.2. Sample timetable.....	35
Table 6.1. Parameter settings .....	53
Table 6.2. Manual and automated timetabling results .....	53
Table 6.3. Average fitness values for hard conflicts.....	56
Table 6.4. Average fitness values for soft conflicts.....	57
Table 6.5. Benchmark results.....	61

**LIST OF SYMBOLS / ABBREVIATIONS**

CSE	Computer Science Department
DNA	Deoxyribo nucleic acid
EA	Evolutionary Algorithms
GA	Genetic Algorithms
GC	Graph Coloring
ITC	International Timetabling Competition
LLE	Linear Linkage Encoding
MOEA	Multi Objective Evolutionary Algorithm
MOGA	Multi Objective Genetic Algorithm
UCT-YU	University Course Timetabling at Yeditepe University
UCT-UU	University Course Timetabling at Udine University

## 1. INTRODUCTION

Searching and finding a solution for a difficult problem is an interesting endeavor and attracts attention of researchers most of the time. As technology developed rapidly, the need for utilizing resources and optimum usage has increased. Developments in computers and computing encourage the research in optimization and search areas. Today, researchers need more and more computing power to perform optimizations and search for optimal or near optimal solutions. Data mining, scheduling, medicine, geometry, bin packing, printed circuit design and frequency assignment can be counted as examples of the optimization areas [1].

Since most of the real-world problems are in the NP-Hard category (i.e. there is no known polynomial time algorithm) many research have been carried out to solve such problems. One of the commonly used methodologies in the area is based on the simulation of the evolution process. In this process nature is observed and then simulated to find solutions [2]. Such methods are classified as Evolutionary Algorithms. Researchers are inspired from evolution to behavior of ants or birds. Genetic Algorithms (GA) are population based biology inspired algorithms [3]. GA is the simulation of nature and biology. This simulation can be as follows: A population is built consisting of a set of individuals. The individuals named as chromosomes are strings consisting of genes. Each gene may take any allele value. In evaluation and fitness assignment stage each individual is evaluated according to an objective function and it is given a "fitness" value. This value shows how "fit" is the individual. This value can also be considered as how far the chromosome is away from the solution. Some individuals are selected for mating and reproduction based on the fitness values. After selection, these individuals (named as parents) are mated and the offspring are created. Crossover is the most common mating process. During crossover, some contiguous part of one parent chromosome is copied into the offspring and the other part of the offspring is copied from the other parent. This process is inspired from genetics: So the hereditary factors are transferred to the offspring. Next, from the offspring pool some genes are selected for mutation. Mutation is the random change in the genes. This provides genetic diversity. Old population and the new

offspring pool are then recombined to form the new generation. As time goes on better and better individuals appear in the process. This cycle continues till the solution is found or a predefined number of iterations has been reached.

Most of the real-world problems are multi objective in their nature. Most of the time, these objectives conflict with each other. Improvements in one objective may harm other objectives and may often yield to undesirable results. So instead of keeping single solution, a set of preferred solutions can be utilized. There are some approaches for this kind; summing objective values and turning problem into single objective one or giving some coefficients for each objective. But these approaches may require fine tuning of coefficients and a high-level of domain knowledge would be needed. One of the techniques used to overcome such problems is the Pareto optimization method. In this technique the set of non-dominated solutions are kept and named as Pareto front. This "elitist" approach combined with Genetic Algorithms yield to NSGA (Non Dominated Sorting Genetic Algorithm) [4].

Scheduling problems are classified as job-shop scheduling, machine scheduling and timetabling. Timetabling problems can be classified as high school timetabling, personal timetabling, university course timetabling and university exam timetabling [5].

University Course Timetabling (UCT) is specifically a constraint-based problem. There are certain rules to be satisfied which are named as hard constraints. On the other hand there are some rules which should be obeyed as much as possible named as soft constraints.

Graph coloring techniques are used for optimization in the areas like medicine, bin packing and scheduling [6]. Timetabling problems can be reduced to a graph coloring problem when the courses are considered as the vertices of a graph and conflicts between the courses are represented as the edges of the graph. In graph coloring, the aim is to color the graph in such a way that no two neighbor vertices have the same color. In UCT case, edges represent the hard constraints. Hence the successful coloring of the graph represents a feasible timetable.

When building timetables, hard constraints must be and soft constraints should be satisfied. As an example two courses cannot be scheduled to same classroom at the same time slot, or same lecturer cannot be scheduled two lectures at the same time slot. Such constraints are named as hard constraints. Reserving a time slot for lunch or minimizing gaps for the lecturers can be named as soft constraints. In a timetable all hard conflicts must be resolved. A feasible and usable timetable is created when all hard conflicts are resolved. In order to have high-quality timetables, soft constraints should be minimized. Our approach is to minimize both hard and soft constraints together.

Since finding a feasible timetable in UCT problem is NP-Hard, there is no known polynomial time algorithm for the solution. In this study, the problem is first reduced to a graph coloring problem [6]. Then Multi Objective Genetic Algorithm (MOGA) framework is applied. Since the problem is multi objective in its nature, a MOGA framework is utilized in this study [4]. Manual Timetabling is an extremely difficult task. A minor change in the table may yield unexpected results. This is due to the fact that all constraints need to be checked again after the update in the timetable. So automation is crucially needed in this area.

In this thesis Yeditepe University Computer Engineering Department's (UCT-YU) timetabling problem is utilized as the first testbed for the multi objective framework proposed. ITC-2007(UCT-UU) benchmark tests are also used in the experiments [18]. These two problem sets consist of university course timetabling problem instances. However, UCT-YU is a real world problem whereas UCT-UU is a semi-real world problem created for benchmarking purposes. UCT-YU is departmental based whereas UCT-UU is faculty or university based. In UCT-UU also room assignment needs to be considered. In UCT-YU case, some constraints exist which cannot be found in the UCT literature. First of all, the timeslots are organized into two different types. The first type of timeslots can only be used for 1-hour lectures. Hence a two hour lecture cannot be scheduled to these timeslots. Certainly the second type of timeslots is reserved for 2-hour lectures. However some of these 2-hour slots can also be used for 1-hour lectures. This constraint effected our representation and design considerably. We needed to map somehow the chromosome into two timetables. Secondly, as a special case, courses in consecutive semesters shouldn't be overlapped. And lastly, practice sessions should

proceed the theoretical sessions. These problems limit possible solutions and affected our tests. In order to solve these problems some specialized operators like group crossover, group mutation are introduced. Also some hybridization is introduced by using some specialized hill-climbers to improve the solutions.

The outline of this thesis is as follows: Chapter 1 is the introduction. In Chapter 2 first a formal definition of the problem is given and then a review of related literature and work about educational timetabling more specifically University Course Timetabling, is presented. There are many kind of approaches already applied to the subject. Among these integer programming, heuristic approaches like simulated annealing, genetic algorithms and multi-phase combination of these methods seem to be the most common ones. Then the features of Yeditepe University Computer Engineering's timetable and University of Udine's timetable for benchmarking are investigated. In Chapter 3 Graph Coloring problem, its definition and applications are investigated. Graph coloring is important in the sense that the UCT is first considered to be a graph coloring problem. The hard constraints are handled as a Graph Coloring (GC) problem in this thesis, too. In Chapter 4 foundations of genetic algorithms are investigated. Natural selection is exposed. Brief information about genetics is given. These two concepts yield to modern Evolution theory. Researchers are inspired from this theory and then these ideas are applied to the field of Computer Science. In this chapter first some basic information about genetics is given. After words, basic concepts of Genetic Algorithms are introduced. Later on, theory of GA is investigated. The end of the chapter is reserved for Multi Objective Genetic Algorithms and Pareto Optimization Techniques. Chapter 5 is devoted to our Multi Objective Genetic Algorithm framework proposed to solve the UCT problem. First design is explained. Then, flow of the algorithm and implementation details are given. Chapter 6 is about the tests done on both datasets used in this thesis. Chapter 7 contains the conclusion and future work.



## 2. TIMETABLING

University course timetabling problem is a constraint optimization problem which consists of three tuples. These tuples are constraints, events and resources. There are two types of constraints in the literature. Constraints that must be obeyed strictly (hard constraints) and constraints should be satisfied as much as possible (soft constraints). Events can be theoretical sessions, practical sessions, recitations and meetings. Resources can be lecturers, assistants, rooms and the facilities in these rooms. University course time tabling is to combine these events with the resources in such a way that all hard conflicts must be resolved and the soft conflicts should be resolved as much as possible. A formal definition of the problem is given in [7] as follows.

“University course timetabling problems (UCTPs) are constraint optimization problems that can be represented by a 3-tuple  $(V, D, C)$ .  $V$  is a finite set of course meetings in a department, faculty or university,  $V = \{v_1, v_2, \dots, v_N\}$ ,  $D = \{d_1, d_2, \dots, d_N\}$  is a finite set of domains of variables, for example, let  $G = \{t_1, t_2, \dots, t_N\}$  represent a set of start times for course meetings, then a possible domain of each variable can be  $d_i \subseteq G$  and  $C$  is a set of constraints to be satisfied,  $C = \{c_1, c_2, \dots, c_L\}$ . Domain of a variable can be a product of sets, each representing a different resource. For example,  $d_i \subseteq G \times S$  can be a domain variable, where  $S$  represents the set of classrooms. UCT can be described as a search for finding the best assignment  $(v_i, t_i)$  for each variable  $v_i \in V$  such that, all the constraints are satisfied. The assignment implies that the course meeting of  $v_i$  starts at  $t_j$  if there are other resources they are allocated for  $v_i$  and starting from that time. Search space is immense  $N^M$ .”

### 2.1. RELATED WORK

There is variety of methods in the literature about timetabling. Some commonly used methods are mathematical programming simulated annealing, tabu search, genetic algorithms, memetic algorithms, constraint satisfaction and neural networks. This list can be extended. In this study integer programming, simulated annealing, genetic algorithms and memetic algorithms are investigated. These methods are applied to high school timetabling, university course timetabling and university exam timetabling.

Integer Programming is applied to solve timetabling problem of University of Patras in Greece [8]. Two sets of binary variables are used basically. These are a set of variables and auxiliary variables.

Basic parameters are defined as:

- $I$  : Day of the week
- $J$  : Time period of the day
- $K$ : Group of students
- $L$  : Lecturers
- $M$ : Courses
- $N$ : Classrooms

One of the constraints is defined as: Lecturers should be assigned at most one course, one group of students and one classroom at a time. This constraint can be formulated as;

$$\forall i \in I, \quad \forall j \in J, \quad \forall l \in L, \quad \sum_k \sum_m \sum_n x_{i,j,k,l,m,n} \leq 1 \quad (2.1)$$

Here  $x_{i,j,k,l,m,n}$  is the basic binary variable which has a value 0 if there is no lecturer assigned in day  $i$  at time  $j$  for group of students  $k$  for course  $m$  and in classroom  $n$ . If this value is 1 then this means only one lecturer is assigned. Formula 2.1 shows that for a specific day, at a specific time period for a specific lecturer when we sum up all  $x$  variables for all group of students, for all courses and for all classrooms the summation should be either 0 or 1. If not then this means there is a conflict related to that specific lecturer. All constraints are formulated like above, and then the part of the objective function might be;

$$\sum_k \sum_l \sum_m \sum_n \sum_i \sum_j c_{i,j,k,l,m,n} \times x_{i,j,k,l,m,n} \quad (2.2)$$

Where  $c_{i,j,k,l,m,n}$  is the cost coefficient of day  $i$  time  $j$  for group of students  $k$  for courses  $m$  and in classroom  $n$ . When all coefficients are the same then all feasible solutions will be optimal. But practically this is not possible and a more specific analysis

for the correct assignments is required. Determining these coefficients may require deep domain knowledge and experience which may not be achievable most of the cases.

Simulated annealing (SA) is an analogy of annealing process of solids. Crystalline solid is heated and then slowly cooled until it reaches its most regular form (i.e. lowest lattice energy state) and thus is free of crystal defects. If cooling is sufficiently slow, the final solid has superior structure integrity. SA is used in such a way to find global minimum. At each iteration two solutions are considered. If the newly found solution improves the current solution, then it replaces the current one. Otherwise it is accepted with the probability given in Equation 2.3. Where  $T$  is the temperature,  $k_B$  is Boltzmann constant and  $\Delta E = E_{cur} - E_{prev}$ .

$$P = e^{\frac{-\Delta E}{k_B T}} \quad \Delta E > 0 \quad (2.3)$$

At high temperatures, the probability  $P$  is very close to 1 and many uphill movements are accepted. As the temperature becomes lower the probability of accepting bad moves decreases and it becomes more difficult to escape local regions of the search space. Abramson et al [9] used this algorithm to build timetables. Fitness value is replaced with energy  $E$ . So the difference in fitness values represent  $\Delta E$ .

Students, courses, lecturers and classrooms are thought as elements. Atoms in the SA process are replaced with the elements of the timetable. Each element is distributed over periods randomly. Then in each iteration, an element is moved from one period to the other. Then the fitness is recalculated since there will be a removal and insertion cost. If the fitness value is improved, the new solution is accepted. Otherwise it is accepted based on the probability distribution mentioned above.

Evolutionary Algorithms (EA) are applied to create feasible and efficient high school timetables in [10]. It is decided not to include courses directly in the chromosome encoding, but instructors are considered in the representation of the chromosome. Chromosome is represented by  $m \times n$  matrix where  $m$  is the number of classes,  $n$  is the number of teaching periods. A value in this matrix represents an instructor assigned at a specific class at a specific time period.

During the initialization phase each instructor is inserted into the timetable. Fitness function decodes the chromosome into a timetable and returns a value representing the fitness of the individual.

Linear ranking selection is used instead of classical roulette wheel ranking selection in this study, since roulette wheel ranking resulted in premature convergence. In linear ranking selection a kind of randomization is included in order to diversify the chromosomes in the population.

Two types of mutations are used: “Period mutation” and “Bad mutation”. In period mutation a row representing classrooms is selected randomly, then two periods are selected and corresponding instructors are swapped for this class. In bad mutation two periods are selected randomly, and then two instructors who have the worst timetables are swapped in that particular chromosome [10].

As an elitist approach, best individual is kept across generations. Other individuals are not kept for the next generation. The degree of elitism is not much here, since only a single individual is kept for the next generation.

University exam timetabling can be also considered as a multi objective combinatorial optimization problem. In [11], it is aimed to disallow students to take exams in consecutive periods or overlapped periods while making the timetable as short as possible without exceeding the seating capacity of a room. Since these objectives are conflicting with each other, a Multi Objective Evolutionary Algorithm (MOEA) is used. Variable length chromosome is used, since fixed length timetables turns the multi objective problem into a single objective one and feasibility is not guaranteed. In this representation each chromosome encodes a complete and a feasible timetable.

Most of the researchers working on educational time tabling problem solvers commented that they do not use crossover since this standard recombination operator produces unfruitful results. In contrast to standard crossover a specialized day-exchange crossover is used in [11].

In day-exchange crossover, only the best days (i.e. the days with less than 3 periods and that have minimum number of clashes per student) are crossed over. After crossing over, a repair process may be needed and the duplicate exams are removed. Again after crossover, the length of the chromosome can be increased, so two more operators are introduced to control the size of the timetables created by the crossover operation. These are period expansion and period packing [11].

Mutation is used to complement the crossover operators to allow larger search space to be explored. For maintaining feasibility, a reinsertion process is used in [11] after the mutation process.

The mutation process includes three different operators. In largest degree mutation, exams with highest conflicts with other exams are reinserted. Color degree, is the same with largest degree, but only it is applied to already scheduled exams. In saturation degree mutation, exams with the fewest valid periods are reinserted. Lastly random degree mutation reinserts exams randomly.

In goal based pareto ranking, two objectives are considered; number of clashes and number of periods of the timetable. Pareto ranking scheme is applied and assigned to variable first rank. If the timetable length exceeds maximum desired length then the difference is added to the first rank to form the second rank, otherwise if its length is less than the minimum length, the difference is added to the first rank to obtain the second rank [11].

For local exploitation, a micro genetic algorithm and a hill climber is used. Micro genetic algorithm solves single objective problem by rearranging the periods. Hill climber is applied to only the best individual. A clash list is maintained and an exam is selected randomly from the list and rescheduled. A strong elitism mechanism is applied by archiving the population [11].

A two phase heuristic evolutionary algorithm is utilized to obtain personalized timetables in Universidade de Vigo [12]. The overall scenario is as follows. The faculty assigns several groups to each subject; each group has a fixed timetable consisting of

previously assigned rooms and laboratories. The groups are already scheduled to staff. All of the students are allowed to register to the subjects that the faculty has provided. Every student makes priority and non-priority choices on subjects and groups. Since each group has a capacity, it's not always possible to do the assignments that students have preferred.

A personalized timetable should include all the priority subjects given to the students and the non-priority subjects should be given as much as possible. If possible, the students should be assigned to the group that they have chosen, if not, they should be assigned to another feasible group.

A two phased heuristic is applied and then in order to improve the quality, a GA is utilized. In the first heuristic, the priority groups are assigned if the current group is full or if there is an infeasible assignment, then priority groups are reassigned. In the second heuristic, the non-priority groups are assigned and if there is an infeasible assignment, the subjects are reassigned. In the GA utilized to improve the quality of the solution, roulette wheel selection with partially matched crossover (PMX) is used [12]. For mutation randomly chosen two strings are swapped.

Lewis and Paechter et al [13] are inspired from GGA (grouping genetic algorithms) and used variety of crossover operators to exploit the relevant building-blocks. Their approach can be summarized as follows, a two dimensional matrix represents the chromosome:  $R \times T$  where  $R$  is the space for rooms and  $T$  is for the timeslots. Each event  $e$  (course or lecture) can be placed anywhere in this matrix. This approach eliminates the hard conflicts since assignment to the same room is prevented.  $L$  is the list of all events. Each event  $e$  attached with a suitable place (room) from the list of rooms  $K$ .

Population is initialized as follows, the event with the smallest  $k$  list length is found in  $L$ . For this particular event a place is chosen. The sum of total number of events in  $L$  feasibly assigned to  $k$  and already assigned events in the same timeslot as  $k$  is calculated and then minimized. Chromosome is updated by inserting event  $e$  into the timetable, other place lists containing  $k$  are updated and  $e$  is removed from  $L$ . These steps are repeated till all events are removed from the list and inserted into the timetable.

The purpose of the new crossover operators utilized in [13] is to discover useful group of genes i.e. building-blocks. Sector based crossover is presented in Figure 2.1. Here *parent1* is copied into *child1* and then a random sector of the *parent2's* genes is copied into *child1*. Of course some events have a chance to be scheduled twice or some events might disappear. So some kind of repairing algorithm is required. Other crossover operators are day based crossover, student based crossover and conflict based crossovers. All defines some region of the chromosome and tries to exploit these useful genes.

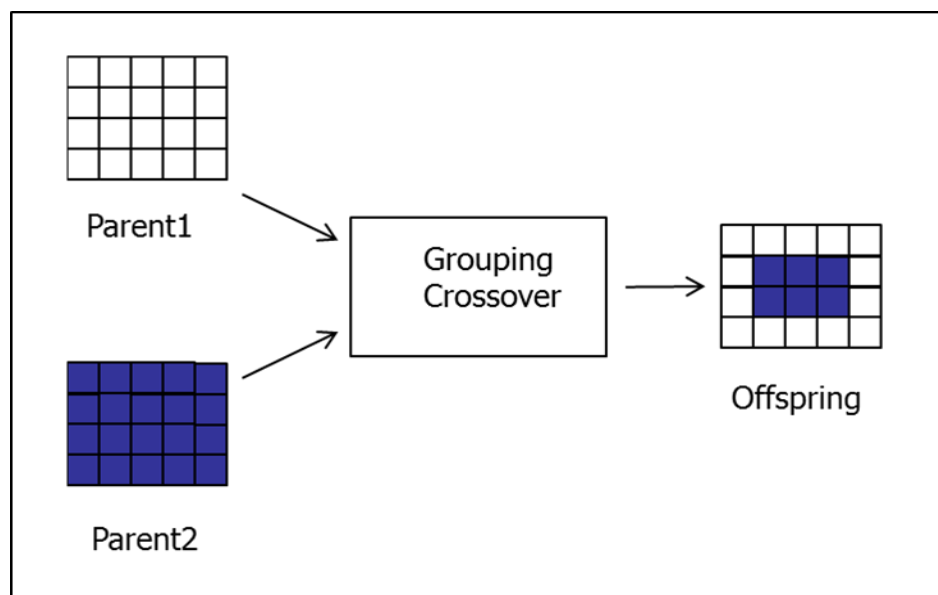


Figure 2.1. Sector based crossover

One of the recent approaches is to combine the evolutionary algorithms with local search algorithms. The resulted algorithm is also called memetic algorithm. As an example, memetic algorithm described in [14] is used to solve university timetabling problem. In this study, hybridization is done using light mutation and then applying randomized iterative improvement algorithm. Selection is roulette wheel selection and mutation is light since it is applied not all but only to a small fraction of the population. Crossover is not used in [14] not to tackle with some extra problems such as repairing the offspring.

In the randomized iterative improvement algorithm utilized in [14], the initial solution is copied as the optimal solution. Then  $K$  number of neighbors is investigated and the best of them is recorded as the local best solution. If local best solution is better than the optimal solution then it is accepted immediately as optimal solution and also copied as current solution. If it does not improve the optimal solution, then optimal is not updated but current solution can be updated with some probability. This probability distribution is defined as;

$$e^{-(f(localbest)-f(current))} \quad (2.4)$$

Hence, there is a chance of accepting non-improving movements as current state in order to escape local minima. In the initialization phase, a solution is obtained by adding or removing the courses to empty the timetable until feasibility is guaranteed. Use of hybridization produced better results than local search on its own [14].

Other than the algorithms presented in this chapter the following algorithms are also utilized. An ant algorithm is utilized for UCT in [15]. Tabu search techniques are used for timetabling [16]. In [17], constraint logic programming is utilized to build timetables.



### 3. GRAPH COLORING

In this chapter first a formal definition of Graph Coloring problem is given and then recent applications of Graph Coloring are investigated. Formal definition of graph coloring is presented in [18] as follows:

“Given a graph  $G = (V, E)$  with vertex set  $V$  and edge set  $E$ , and given an integer  $k$ , a  $k$ -coloring of  $G$  is a function  $c: V \rightarrow 1 \dots k$ . The value  $c(x)$  of a vertex  $x$  is called the color of  $x$ . The vertices with color  $r$  ( $1 \leq r \leq k$ ) define a color class, denoted  $V_r$ . If two adjacent vertices  $x$  and  $y$  have the same color  $r$ ,  $x$  and  $y$  are conflicting vertices, and the  $edge(x, y)$  is called a conflicting edge. If there is no conflicting edge, then the color classes are all independent sets and the  $k$ -coloring is valid. The graph coloring problem is to determine the minimum integer  $k$  (the chromatic number of  $G$  –  $\chi(G)$ ) such that there is a legal  $k$ -coloring of  $G$  [18].”

#### 3.1. RECENT APPROACHES ON REPRESENTATION AND OPERATORS

Graph coloring is a grouping problem. When the evolutionary approach is used on the problem, applying classical genetic operators such as single point crossover or mutation which simply swap two genes may not yield desired results. Also the classical representation schemes have the redundancy problem. This makes the search space enormous.

In order to demonstrate redundancy let's consider the following two individuals encoded with number encoding,  $I_1\{1, 2, 1, 7, 2\}$ ,  $I_2\{8, 5, 8, 9, 5\}$ . In this encoding the value of the gene denotes the color group of the corresponding vertex.

Both of these individuals represent the same solution. They denote the same grouping, but only the color numbers of the groups differ in the individuals. So for grouping problems Linear Linkage Encoding (LLE) scheme is proposed in [6] to remove the redundancy introduced by group number encoding.

In LLE genes in chromosome are represented as links to the other genes. This removes the redundancy mentioned above. LLE scheme and its array implementation are demonstrated in Figure 3.1. According to Figure 3.1 nodes (1,4,6) form a group (have same color), (2) form another group and (3,5) another group.

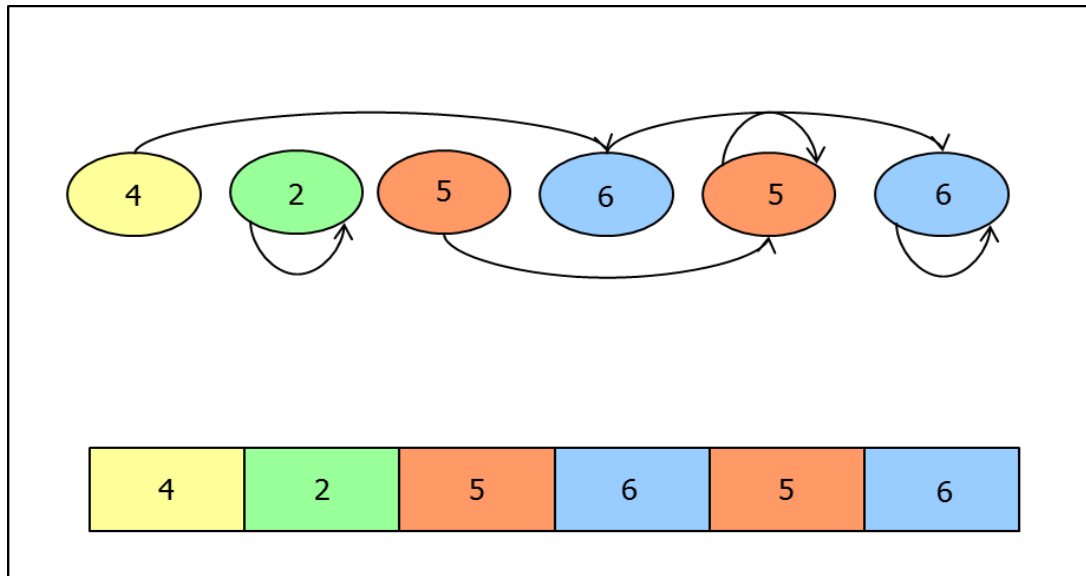


Figure 3.1. LLE and its corresponding array representation

This scheme can also result in redundancy if backward links are allowed or if more than one gene has the same value. So in order to remove this redundancy backward links are not allowed and except the last gene of a group, no two genes allowed to have the same value.

Single point crossover is destructive when grouping problems are considered. If we consider the figure above when single point crossover is applied then the probability of being in the same group for nodes 1 and 6 is 0. But this might be a useful building block close to an optimal solution. Applying single crossover will destroy this block. In uniform crossover there is a possibility that these two nodes can be in the same group, but a specialized operator like group crossover would be more beneficial in order to exploit useful building blocks.

Group crossover is based on the ending nodes since they represent the groups and the number of groups in a chromosome is the number of ending nodes. So this approach puts ending nodes in to the center of this crossover operator. When two parents are considered for crossover the following scheme is proposed.

- Each gene is considered one by one. If the gene is an ending node both in  $P_1$  and  $P_2$  then the gene is accepted as ending node in the offspring created. If it is an ending node in one parent and not in the other, then it is chosen as ending node in the offspring with probability 0.5.
- After the ending nodes are determined, remaining nodes are connected to one of transferred ending nodes again with equal probability. If one ending node is transferred then the current node is attached to that group directly. If both ending nodes are transferred then the node is randomly attached to one of them. If none of the ending nodes are transferred then the node is connected as it is connected in  $P_1$  or  $P_2$ .

However after applying this algorithm we may end up with genes with the same value which contradicts LLE schemes. So some kind of rectification algorithm is required. The details of this rectification process can be found in [6].

LLE-MOGA successfully applied to data clustering on breast cancer and dermatology data sets. LLE-MOGA is also applied to graph coloring DIMACS benchmark suite [6].

There are a number of state of the art algorithms about graph coloring problem. In the graph coloring heuristic proposed in [19] partial solutions and a reactive tabu scheme are used. In this study, partial but feasible solutions are considered and the size of current partial solution is increased by the heuristics utilized. A reactive tabu tenure is also introduced to improve the performance. In [20] a metaheuristic approach is used for vertex coloring problem (VCP) in two phases. In the first phase, evolutionary algorithm is used where in the second phase postoptimization is utilized based on set covering formulation. Memetic algorithms are also utilized and an adaptive multi-parent crossover technique,

together with a distance-and-quality based replacement criterion are also introduced in [21]. A new local search methodology is applied named as Variable Space Search in [22]. In this algorithm search is done according to single objective and when the search sticks in a local optimum then the search is switched to a different objective.

## 4. GENETIC ALGORITHMS

DNA is a nucleoid acid. DNA has pair of molecules in the shape of helical chains which are made from phosphate and sugar [1]. The structure of DNA is shown in Figure 4.1. Helical chains are connected to each other with four types of molecules named as bases. These bases are Adenin, Timin, Guanin and Cytocine. The sequence of these bases encodes the genetic information and whole hereditary information is called as genome [1].

Each cell of an organism contains certain number of chromosomes. As an example goldfish has 100 chromosomes in its cells. Chromosome is an organized DNA and protein structure which is made of many genes. For example *Y* chromosome of human consists of 125 genes and 57740000 total bases. Gene is a locatable region of genomic sequence. The structure of a gene is shown in Figure 4.1. Genes corresponds to biological traits. These traits can be hair color, blood type or a risk for a specific disease. Some of these heritable traits can be useful for individuals for surviving. Then the probability of surviving and reproduction of these individuals increases through generations. This is called as natural selection. There might be a random change in the inherited traits. This change is called as mutation [1].

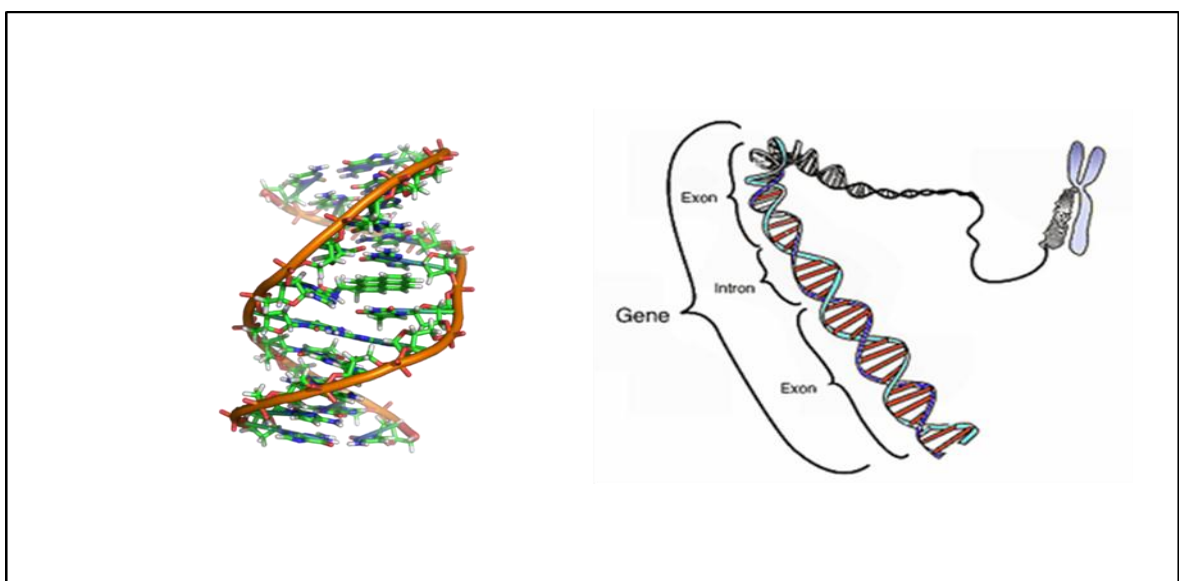


Figure 4.1. Structure of DNA and a gene [12]

## **4.1. GENETIC ALGORITHMS BASICS**

Genetic Algorithms (GA) are population based biology inspired algorithms [3]. GA is the simulation of nature and biology. This simulation can be as follows; A population is built consisting of a set of individuals. The individuals named as chromosomes are encoded strings consisting of genes. The genes have allele values. In the evaluation stage, each individual is evaluated according to some objectives and it is given a "fitness" value. This value shows us how "fit" the individual is. This value can also be considered as how far we are away from the solution.

Some individuals are selected for mating and reproduction based on the fitness values. After selection, these two individuals (named as parents) are mated and the offspring are created. At this stage crossover comes into play. During crossover operation, some part of the first parent is copied into the offspring and the remaining part of the offspring is obtained from the other parent. This process is inspired from genetics; so the hereditary factors are transferred to offspring.

Next, from this offspring pool some genes are selected for mutation. Mutation is the random change in the genes. This provides genetic diversity. Old population and the new offspring pool are then recombined to form the new generation. The more fitter the individual, the more probability it has to survive and to be selected for reproduction. As time goes on better and better individuals are created. This cycle continues till the solution found or a predefined number of iterations has been reached. At the end of the process optimal or near optimal solution is expected to be found [26].

### **4.1.1. Selection**

Selection algorithms can be categorized into two different types. These are selection with replacement or without replacement [3]. In selection with replacement, the individual has a chance to be selected more than once while the individual can be selected only once in selection without replacement. Selection mechanisms are very diverse.

Truncation selection is an example where initially the population is sorted according to the fitness values. Then required number of individuals is selected. This type of selection is a strong elitist approach which cannot preserve diversity.

On the other side, in random selection, the individuals are selected randomly and their fitness is not considered during selection. This scheme provides diversity but this time algorithm may turn into random walk and may not converge to the optimum. Roulette wheel selection balances diversity and elitism in the selection process. In this method, each individual has a chance to be selected with the probability proportional to its fitness over total fitness of the population. Tournament selection is an alternative method where a set of individuals are selected for comparison. Then these individuals compete and the winner of the competition is selected for mating. All of the above selection methods can be utilized with replacement or without replacement.

#### 4.1.2. Crossover

In crossover process, parent chromosomes are combined to form offspring. The most commonly used crossover operation is the one point crossover. In this operation a random position is chosen for crossover, and then as stated before; the part of the offspring up to this crossover point is copied from the first parent and the rest is copied from the second parent. The second offspring is created similarly. Where the roles of the first and second parent are reversed.

As a simple example to demonstrate crossover and mutation operations, let us consider a chromosome of length 6 and let the allele values be in the range 0 – 9. Let's assume the following two individuals are selected as parents  $P_1 = \{ 1, 5, 3, 2, 9, 2 \}$ ,  $P_2 = \{ 4, 2, 7, 1, 6, 8 \}$ . Then let the randomly chosen crossover position be 4, then offspring become:  $O_1 = \{ 1, 5, 3, 2 \mid 6, 8 \}$ ,  $O_2 = \{ 4, 2, 7, 1 \mid 9, 2 \}$ .

Two point crossover is similar to one point, but this time two random positions are selected. For the first offspring middle part is copied from second parent, first and the last parts copied from the first parent. Similarly second offspring is created symmetrically again.

Again let's consider two parents:  $P_1 = \{ 1, 5, 3, 2, 9, 2 \}$ ,  $P_2 = \{ 4, 2, 7, 1, 6, 8 \}$  and let crossover points be 2 and 5 then the offspring created are as follows,  $O_1 = \{ 1, 5 | 7, 1, 6 | 2 \}$ ,  $O_2 = \{ 4, 2 | 3, 2, 9 | 8 \}$

Uniform crossover has a different approach. For each gene allele, the value is randomly chosen either from first parent or second parent. Lets have following parents,  $P_1 = \{ 1, 5, 3, 2, 9, 2 \}$ ,  $P_2 = \{ 4, 2, 7, 1, 6, 8 \}$ .

Let  $P_1, P_2, P_2, P_1, P_1, P_2$  be the sequence denoting the source of allele values chosen for the first offspring and let  $P_2, P_1, P_1, P_2, P_1, P_2$  be the one for second offspring, then the offspring are created as follows,  $O_1 = \{ 1, 2, 7, 2, 9, 8 \}$ ,  $O_2 = \{ 4, 5, 3, 1, 9, 8 \}$ .

### 4.1.3. Mutation

The standard mutation operator changes a randomly chosen gene's allele value. By using mutation it is possible to jump to different parts of the search space and this approach decreases the probability of sticking at local minima.

## 4.2. MULTI OBJECTIVE GENETIC ALGORITHMS

Niched Pareto Genetic Algorithm (NPGA) is introduced as a pareto optimization technique to solve an open problem in hydro systems [24]. The aim is to provide diversity on the front. Since real problems are multi objective in their nature, trying to reduce more than one objectives into a single objective (like giving coefficients or using penalty functions) makes final GA solution very sensitive to the set of coefficients used in the reduction. Therefore, instead of using a single objective and providing a single solution, the utilization of all objectives independent of each other and obtaining a set of solutions can be more preferable. Such an approach is named as Multi Objective Genetic Algorithm (MOGA). In the following paragraphs example algorithms for this framework are presented.



Schaffers [4] VEGA selection is a modified selection method where  $P$  is divided into  $q$  subpopulations size of each  $P/q$  where  $q$  is the number of objectives. Then a new generation is generated by shuffling these subpopulations. This yields to a speciation where some very specialized individuals would be created (according to its own objective function), but there would be little compromised solutions left.

In multi objective genetic algorithms, the domination concept has to be redefined since the individuals have more than one objective in this framework. An individual dominates another one if it is a fitter individual in terms of all objectives used. Certainly, a set of individuals will not be dominated by any other individual in a population. This set is named as the pareto front in the multi objective framework and the genetic process tries to improve this front throughout the search process.

Goldberg [24] proposed non-dominated sorting using rank-selection. According to his algorithm, first non-dominated individuals are identified and given rank 1. Then these are removed and from the remaining population again non-dominated individuals are identified and they are given rank 2. This process continues till there is no individual left.

In Fonseca and Flemings ranking algorithm which is presented in Figure 4.2, each individual is ranked with  $1 + q$  where  $q$  is the number of individuals that dominates the individual [25]. Figure 4.2 demonstrates how these ranks are allocated.

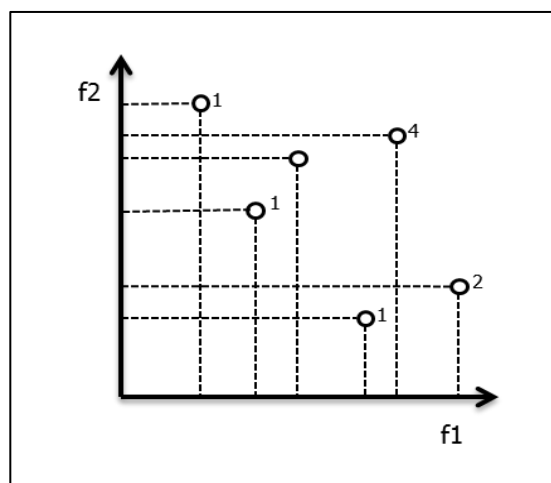


Figure 4.2. Pareto ranking

Fonseca and Fleming used Multi Objective Genetic Algorithms (MOGA) for optimizing the step response of a gas turbine [25]. They utilized a decision making system together with GA. Decision making system is required to determine which individuals will be selected from rapidly growing pareto set. They also used a modified version of pareto ranking.

Goldberg suggested non-domination ranking in [24] and selection in order to move population to non-dominated pareto front and niching mechanism to preserve diversity and preserving convergence to single point on the front. Sharing is used as a niching mechanism shown in Figure 4.3.

Tournament selection is used in this study [24]. An individual is selected from the competing set of individuals. Size of this set defines the selective pressure and convergence speed. Binary tournament exhibit slower convergence and size two is too small for determining the domination rank of the individuals. So first a set of individuals are selected randomly and two other randomly selected individuals are compared with the set. If either one dominates the set it is selected, otherwise in a tie condition sharing is used to determine the winner.

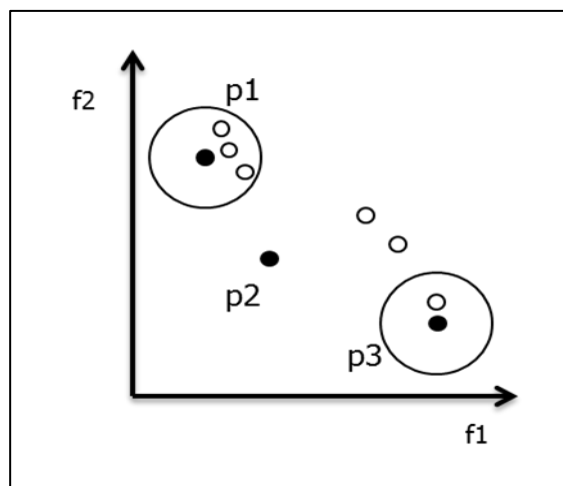


Figure 4.3. Sharing distance

NSGA or similar pareto optimization algorithms are criticized due to their computationally expensiveness and lack off elitist approach. Their algorithm complexity is ( $O(MN^3)$ ) where  $M$  is the number of objectives and  $N$  is the population size. Each solution needs to be compared with every other solution which requires  $O(MN)$  comparisons. To build a non-dominated front  $O(MN^2)$  comparisons are required at this stage. First non-dominated front is formed. Then this first front is temporarily removed and process continues till all fronts are determined. The whole process requires  $O(MN^3)$  comparisons.

NSGA has some drawbacks such as it is a non-elitist approach and it requires user defined qshare parameter to preserve the diversity. Hence NSGA-II is designed to overcome such problems [4].

Among multi objective genetic algorithms MOGA, NSGA and NPGA took much attention. They all share the two common features; non-dominated sorting is used and at the same time front diversity is preserved. Although they have high performance, better approaches are still required in the area. So elitism introduced to enhance convergence.

In SPEA (Strength Pareto EA), an external population is maintained to hold all the non-dominated solutions discovered from the beginning. The external population participates all the genetic operations. At each generation, this external population is combined with current population and fitness's are assigned based on the number of individuals dominated by the individual considered. This assignment directs the search towards the non-dominated ones.

In PAES (Pareto Archived Evolutionary Strategy) one parent and one child is considered. Child is created using mutation operator. If the child dominates its parent, it replaces its parent. If the parent dominates the child, then the child is discarded and a new mutation takes place. If no domination occurs, then decision is made by comparing the archive i.e. the best solutions found so far. Both the parent and the child are compared with the archive. If the child is found to be in a less crowded region then it replaces the parent and also it is added to the archive. Crowding is maintained by dividing the space into subspaces.

Rudolph suggested pareto to form a front by combining the parents with the pareto front of the offspring population to form next generation [4]. This scheme increases elitism and the convergence, but it lacks maintaining the diversity. The non-dominated front can be found using algorithm in Algorithm 4.1.

Algorithm 4.1. NSGA-II finding non-dominated front

$$\begin{aligned}
 &P' = \text{find} - \text{non} - \text{domiated} - \text{front} \\
 &P' = \{1\} \\
 &\text{for each } p \in P \wedge p \notin P' \\
 &\quad P' = P' \cup \{p\} \\
 &\text{for each } q \in P' \wedge q \neq p \\
 &\quad \text{if } p \prec q \text{ then } P' = P' - \{q\} \\
 &\quad \text{else if } q \prec p \text{ then } P' = P' - \{p\}
 \end{aligned}$$

Domination principle used in the algorithm is as follows. An individual  $I_1$  strongly dominates  $I_2$  if it's all objective values are equal to or less than the other. If  $I_1$  dominates  $I_2$  in most of the objective values then this is a loose dominance.

In this algorithm pareto front is initialized with the first individual in the population. Then starting from the second individual, each individual that is not included in the pareto front is processed and it is compared with each individual already contained in the pareto front. If the processed individual dominates any individual in the front, then this dominated individual is deleted from the front and the processed individual is included in the front. Otherwise the processed individual is discarded. And the third case is when there is no domination (i.e. processed individual does not dominate any individual in the front or is not dominated by any of the individuals in the front). In this case the processed individual is included in the front.

As a simple example to demonstrate NSGA-II lets have two objectives and 6 individuals with objective values as demonstrated in Table 4.1.

Table 4.1. Domination Example

Individual	f1	f2
A	4	7
B	3	9
C	5	3
D	3	8
E	7	9

First individual  $A$  would be directly included in the pareto front. Second individual  $B$  would be compared with pareto front (i.e. with  $A$ ) since no domination occurs  $B$  would also be included in the front. When  $C$  is compared with  $B$  and  $A$ , again there is no domination so the front becomes  $\{A, B, C\}$ . When  $D$  is compared with the front it is seen that  $D$  dominates  $B$  so it immediately replaces  $B$ . Hence, the pareto front is formed as  $\{A, C, D\}$ . And the last individual  $E$  is dominated by  $A$ , so it would not be included in the front. Resulting pareto front is,  $\{A, C, D\}$ . In order to find all non-dominated fronts Algorithm 4.2 is used.

Algorithm 4.2. NSGA-II sorting algorithm

```

F = fast - nondominated - sort(P)
i = 1
  while P is not an empty set
    Fi = find - nondominated - front(P)
    P = P \ Fi
    i = i + 1

```

According to this fast-non-dominated-sort algorithm, all fronts are discovered. These fronts are ranked as first non-dominated-front is 1; the second is 2 and so on.

In order to preserve diversity, an estimate for the density of solutions that exists around the solution is required. So average distance to closest solutions on either side is calculated. This can be named as idistance and defines the largest cuboid enclosing point  $i$  without including any other point. This distance is called as crowding distance and it is presented in Figure 4.4. In order to calculate crowding distances Algorithm 4.3 is used.

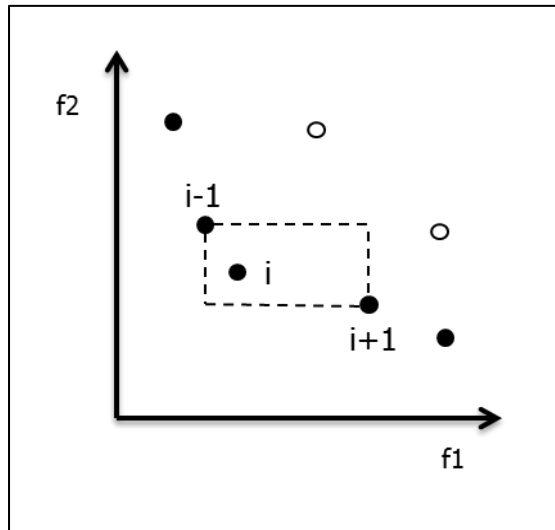


Figure 4.4. Crowding distance

Algorithm 4.3. NSGA-II crowding distance assignment

```

Crowding – distance – assignment( $T$ )
 $n = |T|$ 
  for each  $i$  set  $T[i]distance = 0$ 
    for each objective  $m$ 
       $T = sort(T, m)$ 
       $T[1]distance = T[n]distance = \infty$ 
      For  $i = 2$  to  $(n - 1)$ 
         $T[i]distance = T[i]distance + (T[i + 1]m - T[i - 1]m)$ 

```

Number of solutions in  $T$  is assigned to variable  $n$ . Then all crowding distances are initialized to 0. According to each objective the population is sorted. The boundaries i.e. the first and the last solutions are given infinite values, so that they are always selected. Then the in-between solutions crowding distance is calculated by adding the absolute value of the surrounding objective values. This value is then added to the previous value to cover all objectives and to form the total-crowding distance.

Crowded comparison operator is introduced when two individuals are considered for selection. First their ranks are consulted, the one with lower rank is selected, and if they are at the same rank then crowding distance measures are considered. The one with lower crowded distance i.e. solution in a less crowded area is selected. Now with these operators NSGA-II can be defined as in Algorithm 4.4.

Algorithm 4.4. NSGA-II

```

Rt = Pt ∪ Qt
F = fast – nondominated – sort(Rt)
Initialize Pt as empty set and i = 1
  While |Pt + 1| + |Fi| ≤ N
    Crowding – distance – assignment(Fi)
    Pt + 1 = Pt + 1 ∪ Fi
    i = i + 1
  Sort (Fi, crowded – comparison – operator)
  Pt + 1 = Pt + 1 ∪ Fi[1 : (N – |Pt + 1|)]
  Qt + 1 = make – new – pop(Pt + 1)
  t = t + 1

```

In NSGA-II, the parent and offspring populations are combined to form a temporary population named as  $R_t$ . Then this population is sorted using fast-non-dominated-sort algorithm. Then each non dominated front is included in the next population until

population size is reached. When population size is reached, the current front is sorted according to crowded distance operator. Only the upper portion of this front which does not exceed the population size is included in the next generation.

As a result the overall complexity of NSGA-II is  $O(MN^2)$ . Since solutions compete using crowding distance measures, there is no need such sharing distance parameter demonstrated in Figure4.3. Although crowding distance applied to objective function, it can also be applied to parameter space.

### **4.3. PARETO OPTIMIZATION**

When a multi objective problem is needed to be solved, pareto optimized solutions can be appealing as stated in [5]. Pareto optimal set may provide the set of compromised solutions. In multi objective problems, there is no single best solution. Instead a set of solutions would exist that cannot dominate each other. After the search phase, a decision making process has to be carried out. Even if the best solution is achieved, preferences may change. Objectives may conflict with each other. Satisfying one of the objectives completely may harm other objectives. So a compromised solution is required. Quality of the obtained front depends on; the number of non-dominated solutions, the closeness between the obtained front and the coverage and the distribution of the front. Preserving the diversity of the front is extremely important. Because diversity determines the characteristics of the structure of the solutions that would be achieved.



## 5. MOGA FRAMEWORK

First UCT problems are investigated in this chapter. Then the design of our MOGA framework is explained. At the end of chapter, implementation details are also given.

### 5.1. UNIVERSITY COURSE TIMETABLING

University Course Timetabling problem is a constraint satisfaction problem with three tuples. These tuples are constraints, events and resources. Constraints can be categorized into two: the constraints that must be satisfied strictly and the constraints that should be satisfied as much as possible in order to improve the quality. Events can be theoretical, laboratory, recitation sessions or meetings. There can be a variety of resources such as lecturers, assistants, other staff, classrooms and instruments in the classrooms. The problem can be defined as, events and resources should be combined in such a way that all strict constraints must be satisfied and the other constraints should be minimized.

In this thesis two timetable problems are studied. Although they have common features there are some differences between the two.

1. University Course Timetabling at Yeditepe University Computer Science Engineering Department (UCT-YU) is a real-world problem whereas University Course Timetabling at University of Udine (UCT-UU) is a semi-real-world problem which is a simplified version of the timetabling problem at University of Udine. These simplified instances are created for competition purposes [26].
2. UCT-YU is a departmental based (Computer Science Engineering Department) problem whereas in UCT-UU the whole university or faculty is considered.
3. Both hard and soft constraints differ. In UCT-YU case there are more constraints to be considered compared to UTC-UU. However the instances of UCT-UU are larger.
4. In UCT-YU case two different types of timeslots are present for 1-hour lectures and 2-hour lectures.
5. In UCT-UU the capacity of the rooms is also needed to be considered.

### **5.1.1. Yeditepe University Computer Engineering Department's Time Table**

YU-CSE department has a four year, eight semester program. In this program students have 49 courses in total and they have to complete 149 credits in order to qualify as a computer engineer. During these 4 years students can take three types of courses: general, main and elective courses. General courses are the common courses which should be taken by all students in the engineering faculty. Some example of general courses is Calculus, Physics, Economics, History, Law etc. Main courses are the primary courses to be taken in computer engineering department. Some examples are: Data Structures, Introduction to Programming, Systems Programming and Operating Systems. Elective courses are divided into technical and non-technical courses. Technical ones are further classified as departmental and non-departmental.

Some courses have prerequisites, so without completing the prerequisite student cannot register the course. This is important since prerequisite courses can be overlapped in the timetable to be prepared.

A typical course has a theoretical part and practical part. The practical part consists of laboratory work or recitation. Theoretical course typically have lectures in two different weekdays. In the first day, a 1-hour session and in the second day a 2-hour session is held. A lab session can be 3 hours or 2 hours typically. A recitation is usually a 1 hour session.

CSE department has thirty lecturers including professors, invited professors and research assistants. There are six general purpose labs, six advanced labs, one hardware lab in the department. Around 100 sessions (lectures,lab session and recitation) are held each week.

Timetables in the engineering faculty are created in three phases at Yeditepe University. Faculty schedules the common courses and then submits the initial timetables to departments. Then each department schedules the main and elective courses according to their own preferences. At the last stage faculty assigns the rooms for the courses.

The hard constraints to be obeyed are given below;

1. Lecturer can give only one course at a time.
2. Students of the same semester can participate only one course at a time. So two courses of the same semester cannot be scheduled in to the same timeslots.
3. There are two types of time slots; 2-hour slots and 1-hour slots. The constraint is that, 2-hour slots cannot be allocated to 1-hour sessions (i.e. lectures).
4. 2-hour session (i.e. lecture) and 1-hour session of a course should not be scheduled on the same day.
5. The timeslot between 11:00-13:00 can be allocated as two 1-hour slots or one 2-hour slots but cannot be assigned to both for each day of the week in order to prevent collision. Engineering faculty provides this flexibility hence increases the number of 1-hour slots and 2-hour slots.
6. Some courses like math, physics, engineering management, law have predefined days and slots. These courses are common for different departments. So each department should build its own timetable considering these predefined courses.

The Soft Constraints to be considered are as follows;

1. Courses in consecutive semesters should not be overlapped (exception prerequisite courses). This constraint allows irregular students to take courses from consecutive semesters.
2. Lunch time should be considered. For each day, a timeslot for lunch should be allocated.
3. Elective courses should not be overlapped. This constraint allows students to choose different elective courses in the same semester and provides freedom to choose.
4. Lab courses should follow the theoretical sessions. First theoretical courses are given and then its applications should be studied in the laboratories.
5. Visitor lecturers may have some extra constraints. Since they may have different lectures at different universities, they might be unavailable for certain time periods.
6. Lecturers should have one free day for research. This is the policy in YU Computer Engineering Department.
7. Lecturers should not give lectures for more than four consecutive hours.
8. In a day, lecturers should not have gaps more than four consecutive hours.

9. Departmental meeting should be considered. So during meeting hours there should not be any courses given by any lecturer.

### **5.1.2 University of Udine Benchmarks and ITC2007**

International Timetabling Competition (ITC2007) is the second organization held on the timetabling problem. ITC team aimed to generate new approaches for timetabling within universities. Second important aim for this competition is to combine theory and the practice. Competition is organized as three tracks: The first one is examination timetabling. Students are enrolled on particular courses which have associated exams. The exams have to be scheduled. The second track is post enrolment timetabling. The timetable has to be constructed in such a way that all students can attend the events on which they are enrolled. The last track in the competition is curriculum based timetabling. The timetable to be constructed in this track is based on the curricula published by the university, not on the enrolment data. Although these problems have common features, each one represents a different problem in the educational timetabling field. In this paper ITC2007's curriculum based track's benchmarks are utilized. The hard constraints in these benchmark instances are given below;

1. Lectures: All lectures of a course must be scheduled and they must be assigned to distinct periods.
2. Room Occupancy: Two lectures cannot take place in the same room in the same period.
3. Conflicts: Lectures of courses in the same curriculum or taught by the same teacher must all be scheduled in different periods.
4. Availabilities: If the teacher is unavailable for the period scheduled, than availability conflict occurs.

The Soft Constraints in the same problem set are as follows;

1. Room capacity: For each lecture, the students should fit into the room.
2. Minimum working days: The lectures of a course should be spread into given minimum days.

3. Curriculum compactness: Lectures belonging to a curriculum should be adjacent to each other.
4. Room stability: All lectures of a course should be given in the same room.

## 5.2. MOGA DESIGN

There are a number of design parameters which are utilized in our MOGA. Some of them are the chromosome representation, the population and the fitness function. Chromosome representation is crucial since it defines the search space and the efficiency of the algorithm. Deciding a constant or a variable length population is another issue. Fitness function is also important in deciding whether a single objective or a multi objective algorithm is going to be utilized.

### 5.2.1. Chromosome Representation and Encoding

Chromosome is designed as an array of integers. Number of courses defines the chromosome length. For each course (actually for each lecture, since each course may contain one or more lectures) one index in the array is allocated. Integer number at this index represents the group or the color of the course. Then these groups are mapped to the timeslots in the timetable. This scheme is called as group number encoding and it is presented in Figure 5.1.

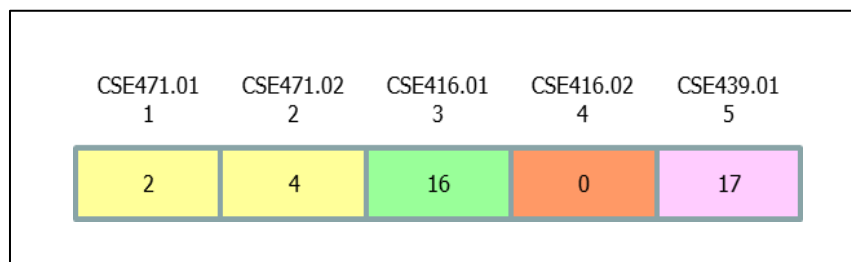


Figure 5.1. Group number encoding scheme

As an example course CSE471 has two sessions. One of them is a 2-hour session CSE471.02 and the other one is a 1-hour session CSE471.01. The number following the course name denotes the session or timeslot type. If it is one then it is 1-hour session and if

it is two then the session is a 2-hour session. The mapping of group numbers into the timetable is as shown in Table 5.1. In this table all 1-hour and 2-hour sessions are listed from Monday to Friday. Hence group number 0 is mapped to the first 1-hour or 2-hour sessions in the table depending on the type of the lecture. Certainly group number 19 is mapped to the last sessions which is Friday 17:00-17:50 for 1-hour sessions and 16:00 17:50 for 2-hour sessions. As an example consider the course CSE471. 1-hour session of course CSE471 which is named as CSE471.01 has a group number 2 as shown in Figure 5.1, this value maps in to hours Monday 13:00-13:50 in Table 5.1, since it is a 1-hour session. This can be seen in the resulting timetable in Table 5.2. Similarly CSE471.02 has group number 4 and this value maps in to hours Tuesday 09:00-10:50 since it is a 2-hour session. Again this can be seen in the timetable in Table 5.2.

Table 5.1. Group number timeslot mapping

Group number	1 hour timeslot	2 hour timeslot
0	Monday 11:00-11:50	Monday 09:00-10:50
1	Monday 12:00-12:50	Monday 11:00-12:50
2	Monday 13:00-13:50	Monday 14:00-15:50
3	Monday 17:00-17:50	Monday 16:00-16:50
4	Tuesday 11:00-11:50	Tuesday 09:00-10:50
5	Tuesday 12:00-12:50	Tuesday 11:00-12:50
6	Tuesday 13:00-13:50	Tuesday 14:00-15:50
7	Tuesday 17:00-17:50	Tuesday 16:00-16:50
8	Wednesday 11:00-11:50	Wednesday 09:00-10:50
9	Wednesday 12:00-12:50	Wednesday 11:00-12:50
10	Wednesday 13:00-13:50	Wednesday 14:00-15:50
11	Wednesday 17:00-17:50	Wednesday 16:00-16:50
12	Thursday 11:00-11:50	Thursday 09:00-10:50
13	Thursday 12:00-12:50	Thursday 11:00-12:50
14	Thursday 13:00-13:50	Thursday 14:00-15:50
15	Thursday 17:00-17:50	Thursday 16:00-16:50
16	Friday 11:00-11:50	Friday 09:00-10:50
17	Friday 12:00-12:50	Friday 11:00-12:50
18	Friday 13:00-13:50	Friday 14:00-15:50
19	Friday 17:00-17:50	Friday 16:00-17:50

Table 5.2. Sample timetable

		MONDAY	TUESDAY	WEDNESDAY	THIURSDAY	FRIDAY
1	09:00 09:50	CSE416.02	CSE471.02	CSE439.02	CSE344.02	
2	10:00 10:50	CSE416.02	CSE471.02	CSE439.02	CSE344.02	
3	11:00 11:50				CSE421.01	CSE416.01
4	12:00 12:50					CSE439.01
5	13:00 13:50	CSE471.01	CSE488.01			
6	14:00 14:50			CSE488.01		
7	15:00 15:50			CSE488.01		
8	16:00 16:50		CSE442.02		CSE344.02	
9	17:00 17:50		CSE442.02		CSE344.02	
10	18:00 18:50			CSE478.01		

Chromosome encoding is an important design parameter. As mentioned before variable length and fixed length approaches are present in the literature. Choosing a right representation scheme is extremely important since wrong choices may yield enormous search space. One dimensional or two dimensional array representations are also possible. In two dimensional cases while one dimension represents the slots or the periods, the other dimension would represent the classrooms. Each cell of the two dimensional array would represent either the event or the lecturer. In this study, group number encoding scheme is used. In order to reduce redundancy and increase search efficiency other encoding schemes like LLE are present [6]. LLE is used in grouping problems and well applied to medicine and bin packing [6]. In our study we utilized group number encoding instead of LLE since it is simpler to utilize and does not require a repairing algorithm.

A single dimensional array representation in UCT-YU case is sufficient for the group number encoding scheme utilized. A second chromosome is needed to allocate the rooms to courses in UCT-UU case. Each gene's value designates a room. For example if the gene has value 1, the corresponding course would be scheduled to room number 1. The room scheduling that exists in second chromosome is evaluated in terms of the hard constraint room availability and soft constraints which are room capacity and room stability in UCT-UU case.

Fixed length chromosome is used in order to guarantee that all the sessions are scheduled. This scheme directly resolves the first hard constraint which denotes that all lectures should be scheduled.

### **5.2.2. Population**

There are many approaches for representing population. Fixed or variable sized populations can be used in a multi objective framework. In the literature, population size is recommended to be half size of the chromosome length. Variable sized populations are used due to fact that; the number of non-dominating solutions increases with time, so when all of them are included in the population, the size of the population should also be increased. But this approach might be computationally inefficient. Although this approach extends the search space, there might be a convergence problem and the search can become more randomized. In order to overcome such problems a fixed size of a achieve population is used in this study. This is an elitist approach and may result in fast convergence. Some algorithms combine current population and offspring populations to build next generation (transitional GA) [4]. Some of them used mating pool of individuals in order to achieve mating and crossover processes.

Fixed size population algorithms try to improve the solutions by pruning the optimal front and the non-dominated solutions. Clustering techniques are also used in order to prune pareto optimal solutions.

In our study fixed size of population is used. Although half of the chromosome length is recommended as population size, lower population sizes like 20 gave reasonable results in the experiments.



### 5.2.3. Fitness Function

Let  $T$  be an individual than the objective is to minimize the fitness function  $F(T)$ .  $F(T)$  is composed of  $H(T)$  and  $S(T)$  which are fitness functions for hard and soft conflicts. These functions can be further divided in to sub-objectives or dimensions and can be represented as  $h_i(T)$  and  $s_j(T)$ . Each  $h_i(T)$  represents the fitness function for each dimension of hard conflicts. Similarly each  $s_j(T)$  represents the fitness function for each dimension of soft conflicts. Here  $n$  represents the total number of dimensions or objectives for hard conflicts and  $m$  represents the total number of dimensions or objectives and for soft conflicts. Then our objective is to minimize all of these objectives as stated in Equation 5.1.

$$\text{Minimize}\{h_1(T), h_2(T) \dots h_i(T) \dots h_n(T), s_1(T), s_2(T) \dots s_j(T) \dots s_m(T)\} \quad (5.1)$$

The first hard constraint denotes that all lectures of a course must be scheduled. A violation occurs if a lecture is not scheduled. This constraint is directly satisfied by dedicating a gene for each lecture in the chromosome representation. Let the following variables denote the following values.

- $I$  : Day of the week
- $J$  : Time period of the day
- $K$ : Group of students
- $L$ : Lecturers
- $M$ : Courses
- $N$ : Classrooms

Second hard constraint is two lectures cannot take place in the same room in the same period and can be formulated as follows;

$$\forall i \in I, \quad \forall j \in J, \quad \forall n \in N, \quad \sum_m \text{roomocp}_{i,j,m,n} \leq 1 \quad (5.2)$$

And the corresponding fitness function is;

$$h_2(T) = \sum_n \sum_m roomocp_{i,j,m,n} \quad (5.3)$$

where  $roomocp_{i,j,m,n}$  variable is 1 if the room  $n$  is reserved for room  $m$  on day  $i$  and at timeslot  $j$ , otherwise its value is 0.

Third hard constraint is as follows, lectures of courses in the same curriculum or taught by the same teacher must all be scheduled in different periods. Hence the fitness function can be defined as

$$h_3(T) = \sum_l \sum_m lec_{i,j,l,m} + \sum_k \sum_m cur_{i,j,k,m} \quad (5.4)$$

where  $lec_{i,j,l,m}$  variable is 1 if the lecturer  $l$  of course  $m$  has a lecture on day  $i$  and at timeslot  $j$ , otherwise its value is 0. Variable  $cur_{i,j,k,m}$  has a value of 1 if curriculum  $k$  has course  $m$  on day  $i$  at timeslot  $j$ , otherwise its value is 0.

Forth hard constraint is, if the teacher is unavailable for the period scheduled, than availability conflict occurs.

$$h_4(T) = \sum_l \sum_i \sum_j unav_{i,j,l} \quad (5.5)$$

where  $unav_{i,j,l}$  variable is 1 if the lecturer is unavailable on day  $i$  and at timeslot  $j$  and has a course scheduled on day  $i$  and at timeslot  $j$ , otherwise its value is 0.

First soft constraint is the room capacity constraint. For each lecture, the students should fit into the room.

$$s_1(T) = \sum_m Cap(m, n) \quad (5.6)$$

where  $Cap(m, n) = NumStud_m - Cap_n$  if  $NumStud_m > Cap_n$ , otherwise  $Cap(m, n) = 0$ . Capacity function returns the difference between number of students scheduled for course  $m$  and the capacity of that particular room. If capacity is higher than the number of students than this function returns 0.

Second soft constraint is the minimum working days constraint. In this constraint the lectures of a course should be spread into given minimum days.

$$s_2(T) = 5 \times \sum_p MinW_p - G(p) \quad (5.7)$$

where  $p$  is the course group number of the lectures.  $G(p)$  returns the number of days the course  $p$  spread in to.  $G(p) < MinW_p$  then above formulation is applied, otherwise value zero is used for  $s_2(T)$ .

Third soft constraint is the curriculum compactness. In this constraint lectures belonging to a semester should be adjacent to each other.

$$s_3(T) = 2 \times \sum_k \sum_b (1 - (H(S_{k,b+1}) - H(S_{k,b}))) \quad (5.8)$$

where  $S_{k,b}$  is the  $b$ 'th course at semester  $k$  and  $H(S_{k,b})$  returns the group number of the course  $S_{k,b}$ .

Forth soft conflict is room stability. Room stability constraint can be defined as all lectures of a course should be given in the same room.

$$s_4(T) = \sum_p \sum_m (ocp_{p,m} - 1) \quad (5.9)$$

where  $\sum_m (ocp_{p,m} - 1)$  is the number of different rooms reserved for course group  $p$ .

### 5.3. FLOW OF ALGORITHM

The flow of algorithm is demonstrated in Figure 5.3. First population is initialized randomly. Then timetables are created, the fitness values are assigned to the individuals and then pareto front is initialized. Next individuals are selected for mating process. Crossover and mutation operators are applied to these selected individuals to form offspring pool. Later on hill climbers are utilized. Current generation and offspring are combined to form the next generation. This process is continued till the solution is found or a predefined number of iterations has been reached.

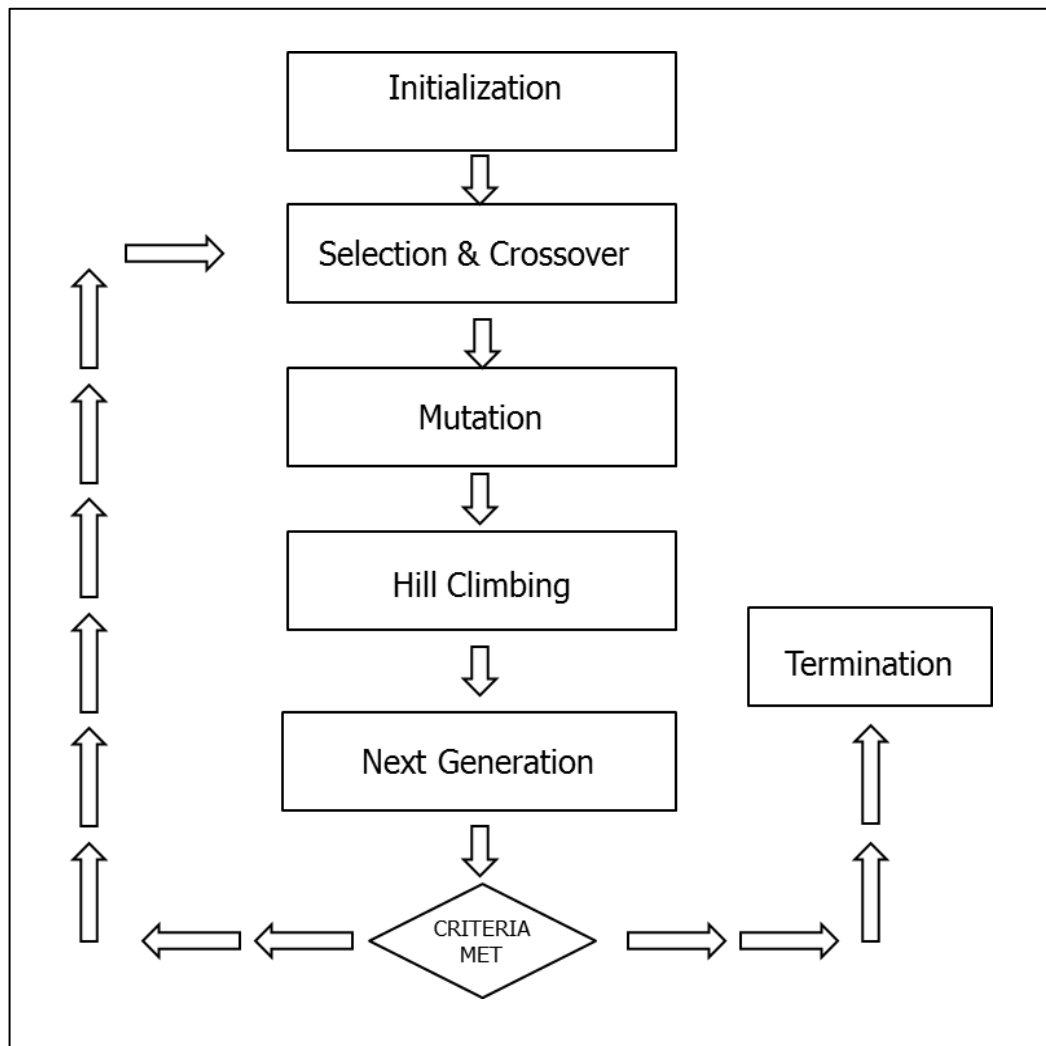


Figure 5.2. Flowchart

### 5.3.1. Initialization and evaluation

Population is initialized randomly. Each chromosome initialized providing a seed to the random number generator. For each gene in the chromosome, a random value between 0 and number of colors (i.e. number of timeslots is 20 in UCT-YU case) is assigned. Similarly room chromosome is initialized. For UCT-UU case, for each gene in the room chromosome, a random value between 0 and number of rooms is assigned. So if the gene in the room chromosome has a value 3, then this means that the room 3 is assigned for the corresponding course.

After the timetables are built, each individual's fitness value is evaluated. Then pareto front is initialized using NSGA-II [16] non-dominating sorting algorithm in Algorithm 5.1. This step is required since after the reproduction stage, the new generation is formed and the first pareto front needs to be updated. In this algorithm pareto front is initialized by directly including first individual in the population. Then starting from the second individual, each individual that is not included in the pareto front is processed and compared with each individual already contained in the pareto front based on their fitness values. If the processed individual dominates any individual in the front, then dominated individual is deleted from the front and processed individual is included. Otherwise, if the processed individual is dominated by an element in the front, it is not included in the front. And the third case is when there is no domination (i.e. processed individual doesn't dominate any individual in the front or is not dominated by any of the individuals in the front). In this case, the processed individual is again included in the front.

Algorithm 5.1. NSGA-II finding non-dominated front [16]

```

P' = find - nondominated - front
P' = {1}
for each  $p \in P \wedge p \notin P'$ 
     $P' = P' \cup \{p\}$ 
for each  $q \in P' \wedge q \neq p$ 
    if  $p \prec q$  then  $P' = P' - \{q\}$ 
    else if  $q \prec p$  then  $P' = P' - \{p\}$ 

```

### 5.3.2. Selection

Individuals are selected for mating. In roulette wheel selection each individual has a chance to be selected proportional to its fitness value. In tournament selection, a set of individuals conforms the comparison set and each individual to be selected is compared to

the elements of this set and then they are selected. Binary tournament is the special case where this comparison set is conformed by only two individuals to be selected. In our MOGA framework instead of classical roulette wheel selection two different types of tournament selection algorithms are utilized. The first one is the standard binary tournament selection algorithm (BTS). The second one is a modified version of binary tournament selection similar to selection algorithm in [4]. This is named as binary tournament with crowding distance (CDS) in this study. In this algorithm, ranks of individuals and their crowding distance values are determined. Rank of the individual is simply the number of individuals it is dominated by. This type of rank assignment is different than in [4]. In that algorithm, ranks are assigned according to the front that the individual is in. if the individual is in the first front then the rank value 0 is given, if it is on the second front then the rank value 1 is given and so on. Crowding distances are calculated as follows:

Algorithm 5.2. Crowding distance assignment

*For each objective*  
*Sort the population based on fitness values*  
*For each individual*  
*If the individual is the first or the last one after sorting*  
*Set Crowdistcur as the infinite value*  
*Else if the individual is an intermediate one*  

$$Crowdistcur = Crowdistcur + Fitness_{prev} - Fitness_{next}$$

This calculation is different than the one in [4]. These distances are applied to each front separately in [4] but the whole population is used for calculations in this thesis.

In the selection, with or without replacement selection can be used [3]. In this study with replacement scheme is chosen. This provides re-selection of the already selected individual for mating. If with replacement algorithm is chosen selective pressure increases,

this may result in fast convergence, but at the same time there is the risk of premature convergence. If without replacement algorithm is used, then diversity can be achieved, but with the possibility of random walk. In this work with replacement algorithm is preferred since an effective convergence is the main concern to obtain feasible timetables. Individuals selected as parents are crossed-over based on a crossover rate. Part of the offspring population is built upon the generated offspring by crossover operation and the other part is formed by copying the individuals from the current population.

### 5.3.3. Crossover

Crossover is done at crossover rate and applied to selected parents. The crossover rate is the rate at which individuals are selected for mating. As a common rate in the literature crossover rate is selected as 90% in this study. So when an offspring pool is to be formed, 10% of the individuals are selected randomly and copied as they are, and the remaining 90% is formed by the crossover operation. Three crossover schemes are utilized and tested in our MOGA which are, Single Point Crossover (SX), Uniform Crossover (UX) and Group Crossover scheme (GX). The standard GX is proposed in [18]. However in this study, this scheme is combined with UX to exploit the useful building blocks but at the same time to preserve diversification. Since these objectives are conflicting with each other, they need to be balanced. The new group crossover utilized in this study is defined in Algorithm 5.3.

Algorithm 5.3. Group crossover algorithm

*Sort the colors according to their group fitness values*

*Select the worst color of parent1 and copy it from parent2*

*Select the worst color of parent2 and copy it from parent1*

*Randomly distribute the courses with worst color of parent1*  
*that are not in parent2*

*Randomly distribute the courses with worst color of parent2*  
*that are not in parent1*



With this algorithm, it is assumed that the overall fitness will improve since we are destroying the worst group and trying to exploit the building blocks on the other parent. The probability of getting a better individual is high since the destroyed group is the worst one and the copied group from the other parent is probably a better one.

#### **5.3.4. Mutation**

Mutation is applied to offspring population at mutation rate. As a common rate in the literature this rate is set to  $1/\textit{chromosome\_length}$  in this study. Hence, for each chromosome, a single gene is expected to be mutated. Single mutation and group mutations are used. All offspring has a chance to be mutated. In literature, the other possibility is to apply mutation only to copied offspring but not to the crossed-over offspring. The mutational operators used in the framework are as follows;

Mutate<sub>single</sub> operator affects a single gene value of a chromosome. It randomly chooses a gene of a chromosome and changes its allele value with a new random value.

Mutate<sub>group</sub> operator affects a group of genes. Two random colors/groups are chosen and they are swapped in the chromosome.

Mutate<sub>group-combine</sub> operator affects a group of genes. Here, three colors are selected randomly. The genes of the first two colors are sent to the group represented by the third color with probability 0.5. Hence some of the genes will be kept in their original colors.

Mutate<sub>group-distribute</sub> operator also affects a group of genes. Three colors are selected randomly. The genes of one color are distributed randomly to the groups represented by the other two colors. Again, the genes can stay in their original group with probability 0.5.

### **5.3.5. Hillclimbing**

Hillclimbing is also applied to the offspring created at each generation. The hillclimbing operators used in this study are explained as follows:

**Hillclimb-single-both:** An individual's neighborhood is searched through iterating single gene's value. If there is an improvement in both hard and soft constraints the individual is replaced with the new one immediately. Otherwise, the old individual is kept and iteration is continued with a different gene.

**Hillclimb-single-hard:** Same process as hillclimb-single-both is applied to the individual but this time improvements only in hard constraints are considered.

**Hillclimb-single-soft:** Same process as hillclimb-single-both is applied to the individual but this time improvements only in soft constraints are considered.

**Hillclimb-group-both:** A new individual is formed by randomly choosing a group mutation and then applying it to current individual. Then this newly created Individual is compared with the current individual to check if there is an improvement in all both hard and soft objectives. If this is the case the current individual is replaced with the new one, otherwise the same procedure is applied to the current individual once more.

**Hillclimb-group-hard:** Same as hillclimb-group-both except only hard constrained objectives are taken into account.

**Hillclimb-group-soft:** Same as hillclimb-group-both except only soft constrained objectives are taken into account.

### **5.3.6. Next Generation**

Current population and the offspring population are combined to form the population for next generation. A number of recombination schemes are utilized and tested in this study. In the first and second recombination schemes, pareto front is not considered. In

third, fourth and fifth schemes pareto front is used to help convergence as an elitist approach. In all schemes, each individual in current population is paired with its peer in the offspring pool. This is due to fact that peer offspring is either the mutated or crossed over version of the initial individual. Hence it will be similar to the original one.

In the first recombination scheme utilized, each offspring is compared with its peer individual and replaces this peer individual if the offspring dominates it. In the second scheme, the offspring replaces its peer individual, if it is not dominated by this peer individual. This second scheme significantly improved the solutions obtained in the experiments. Third scheme introduces elitism, first pareto front part of the current population and its peer part of offspring pool is considered. Each offspring is compared with its peer individual in the current population which would be in the pareto front. If the offspring dominates the individual in the pareto front, the offspring immediately replaces it. Otherwise pareto front is preserved. On the non-pareto front part of the population offspring not dominated by its peer individual replaces the current peer individual. This elitist scheme improves convergence and hard fitness values, but soft ones become worse. Fourth scheme is same as third scheme except that each individual dominated by its peer offspring are reinserted into non-pareto front part of the population. This scheme improved both hard and soft fitness values. Fifth scheme is the same as forth one except that the non-dominating offspring (peer to the original pareto front) are reinserted in to non-pareto front part of the population. Convergence and diversification is further improved by trying to reinsert such offspring comparing them with the rest of the non-pareto front part of the population. Each such offspring is kept in this part of the population if it dominates any of the individuals in this non-pareto front part. Reinserting these non-dominating offspring into the next generation provides diversification and better search paths.

A numeric example is given as follows and demonstrated in Figure 5.3. Current population is consists four individuals  $I_1, I_2, I_3$  and  $I_4$ . After selection  $I_1 - I_3$  and  $I_2 - I_4$  are selected as parents. Applying crossover operator creates offspring population at crossover rate which is 90%. After applying mutation some offspring have changed at mutation rate which is 0.008. Applying these operators forms offspring population. Then next generation is built by applying next generation algorithm. Let's assume that  $I_1$  and  $I_2$  forms pareto front,  $O_{11}$  dominates  $I_1$ ,  $I_2$  dominates  $O_{22}$ ,  $I_3$  dominates  $O_{33}$  and  $O_{44}$  is not

dominated by  $I_4$  when their fitness values are considered. Since  $O_{11}$  dominates  $I_1$ ,  $O_{11}$  repla  $I_1$  in the next generation. Since  $I_2$  dominates  $O_{22}$  remains in the population. Similarly  $I_3$  is survived in the next generation since it dominates  $O_{33}$  and  $O_{44}$  is replaces  $I_4$  since the offspring is not dominated by  $I_4$ .

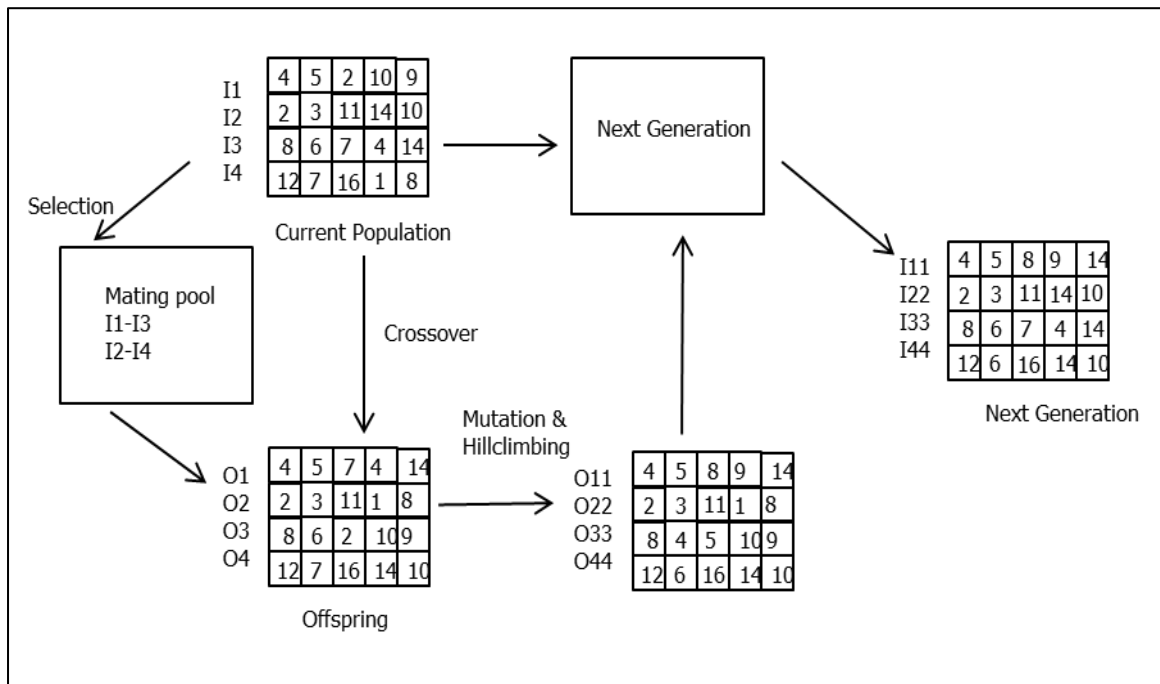


Figure 5.3. MOGA example

## 6. TESTS

The general methodology of testing is as follows. Mutation rate is set to  $1/\text{chromosome\_length}$  for all experiments conducted. In order to determine the optimum configuration for the parameter set, the following approach has been used. The tests are carried out first with a random configuration, then a parameter is modified and result is checked to see whether there is an improvement or not. If there is no improvement the modification is taken back and another parameter is modified. If the modification improves the current configuration then current configuration is replaced by the modified configuration. There are number of parameters that needed to be tested. These are population size, multi objectivity level, pareto size, crossover rate, the amount of guided search and the effect of hill climbers. Some of these parameters are investigated on Yeditepe University CSE UCT testbed and some of them are investigated on benchmark testbeds.

### 6.1. YEDİTEPE UNIVERSITY CSE UCT TESTS

In the experiments, initially the effect of population size is tested on UCT\_YU dataset. Normally population size is recommended as half of the chromosome size in literature. But smaller population sizes yield better results in the testbeds used. In Figure 6.1 and Figure 6.2, best fitness values averaged over 10 runs for different population sizes are demonstrated. Each run used in these experiments lasts for 100 seconds. Figure 6.1 shows hard fitness values and Figure 6.2 shows soft fitness values. Population size of 20 performs the best among all, so it is chosen for the rest of the tests.

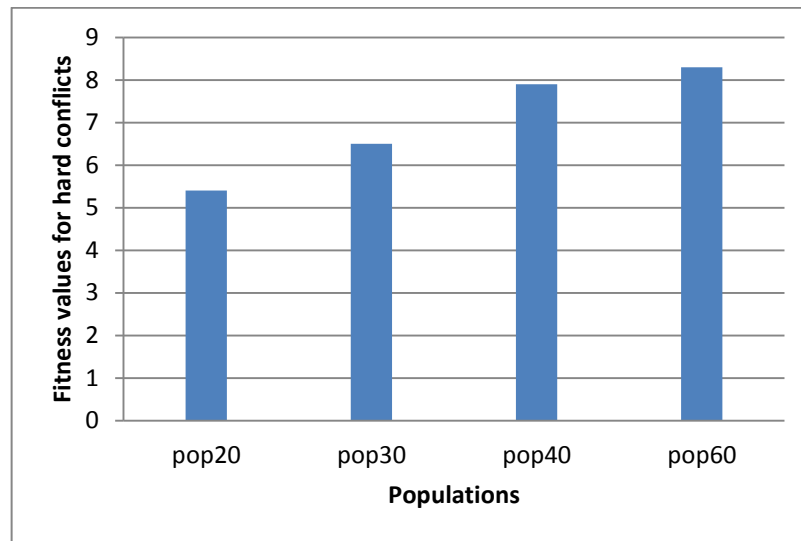


Figure 6.1. Comparison of population sizes for hard conflicts

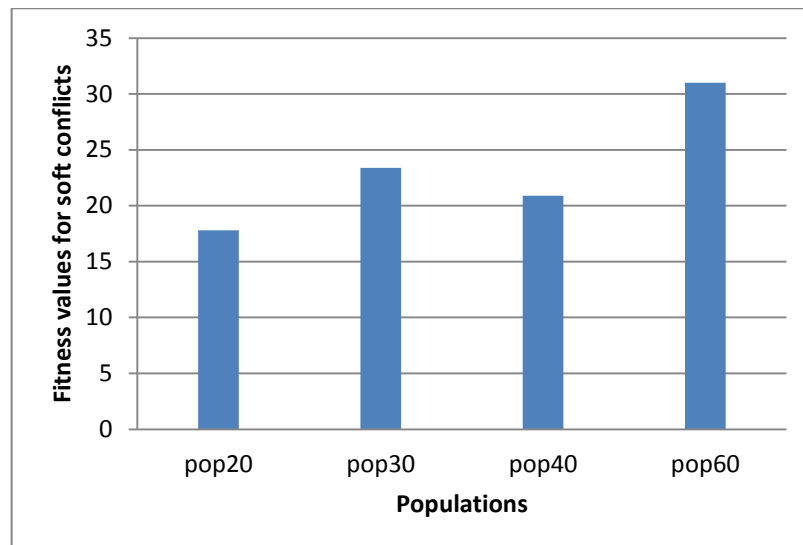


Figure 6.2. Comparison of population sizes for soft conflicts

Although the UCT problem is a multi objective one, by summing the weighted fitness values the problem can be reduced to a single objective one and standard GA can be utilized to solve it. For this reason, different levels of multi objectivism are tested to see if an improvement can be achieved with the multi objective framework. When all hard and soft constraints are summed up, a single objective framework is achieved. Multi objectivity

level can be increased by first dividing this single objective scheme into two objectives, hard conflicts and soft conflicts. Then these objectives can be further divided into sub objectives. This can be achieved by further categorizing the objectives. For example soft objectives can be divided into two objectives by combining room capacity and minimum working days together and combining curriculum compactness and room stability together. When we consider each constraint as a separate objective then we would have maximum multi objectivity. For the soft constraints room capacity, minimum working days, curriculum compactness and room stability would then be separate objectives. The same approach can be used to determine the multi-objectivity for the hard constraints.

Increasing multi objectivity level improved the solutions in the experiments of UCT-YU instance. This is an expected result since more diverse individuals are kept in the pareto front when multi objectivity level is increased. In Figure 6.3, the average values of 10 best solutions over 500 generations for each framework are shown. The one with 3 objectives begins to perform better than single objective one after 400 generations. The one with 4 objectives begins to perform better than single objective one after 350 generations. The one with 7 objectives begins to perform better than the 5 objective one after 300 generations. When we consider the success rates, single objective framework ended with 3 hard conflicts in 70% of the runs and ended with 2 hard conflicts in 30% of the runs. Tests with 7 objective functions ended with 2 conflicts in 30% of the runs, ended with 1 conflict in 40% of the runs and ended with no conflict at all in 30% of the runs. These figures show that, multi objectivity performs better then single objective framework in UCT-YU case. For UCT-UU case increasing multiobjectivity level after some point did not improve the solutions. So two objectives for hard conflicts and two objectives for soft conflicts (four objectives in total) are utilized in UCT-UU instances.

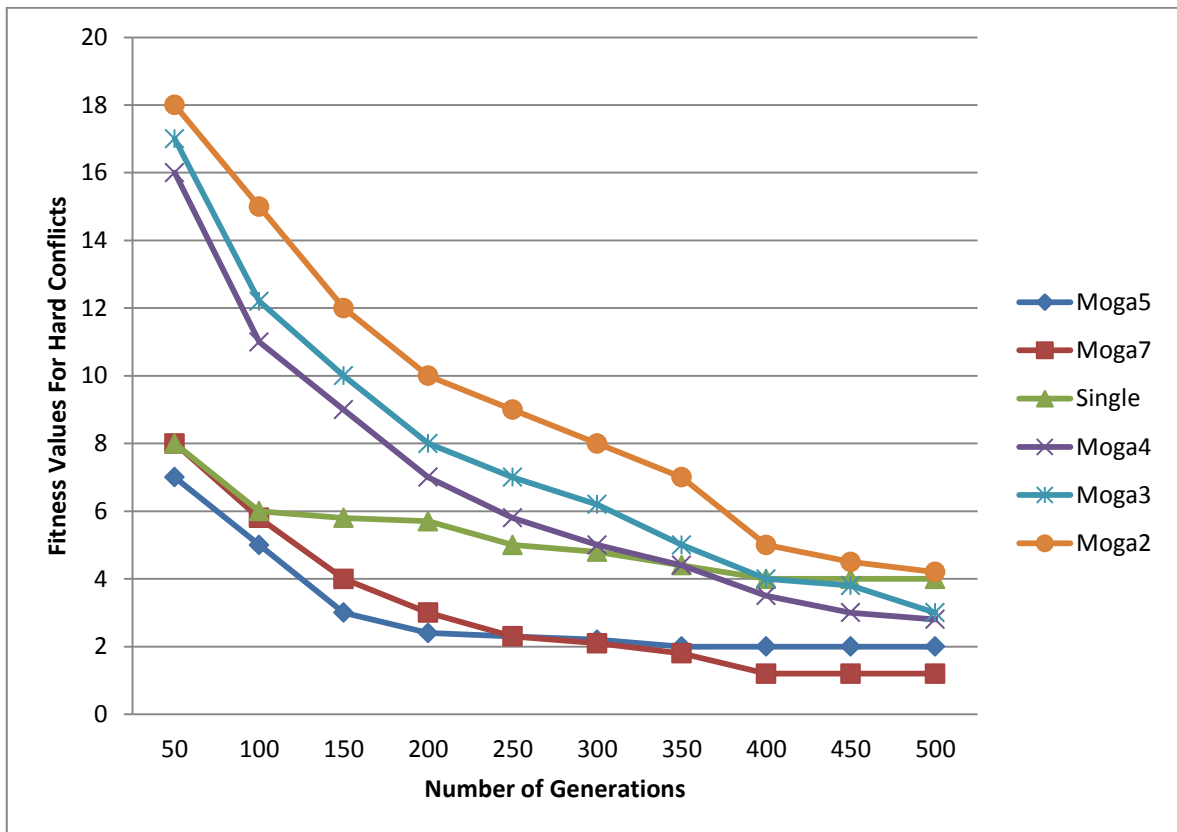


Figure 6.3. Single vs. Multiobjective

Pareto size is one of the important parameters. Pareto front is the fixed portion of the population which consists of non-dominated elements. When this parameter is increased, convergence and fitness values for hard conflicts improve, but the soft ones become worse. Decreasing pareto size improves the fitness values for soft conflicts but this time hard ones become worse. This seems reasonable since increasing elitism resulted in faster convergence and decreasing elitism resulted in diversification but slow convergence. Since the size of the pareto front is fixed, a pruning mechanism is needed. A number of mechanisms are utilized. First the closest pair of elements from pareto front are chosen as victims. Then one of them is pruned with probability 0.5. Second, sharing distance values of the pareto front are calculated. Since this variable represents the density around that individual, the one with the highest sharing distance value may have many similar individuals around. Hence this individual can be pruned. In the last scheme named as crowding distance assignment, the individual with the lowest crowding distance value is



the candidate for pruning process. Pareto size is selected as 50% of the population in the experiments, since it provides a balance between the hard and soft objectives.

*Crossover/copy* rate is also an important parameter. Crossover rate is the portion of offspring pool which is formed by crossover operator. Copy rate is the portion of offspring pool which is directly copied from current population. All of the offspring pool can be mutated. Normally 90% crossover rate 10% copy rate is recommended in the literature. In most of the experiments, it has been observed that setting these parameters as 90% – 10% produces the best results.

After these tests the parameters that are shown in Table 6.1 are used for the UCT-YU problem set and the result in Table 6.2 is obtained. As shown in Table 6.2 automated timetabling has a much higher quality than the manual timetabling.

Table 6.1. Parameter settings

Parameter Name	Value
Population	20
ChromosomeLength	125
CrossoverRate	0.90
CopyRate	0.10
Mutlrate	0.008
paretosize	0.5

Table 6.2. Manual and automated timetabling results

Man-made timetable Hard conflicts	Man-made timetable Soft conflicts	MOGA timetable Hard conflicts	MOGA timetable Soft conflicts
0	45	0	7

## 6.2. BENCHMARK TESTS

The guided search, hill climbers, different selection and crossover schemes in order to improve the performance and quality of the timetables are investigated through the benchmark instances of UCT-UU. Although the tests are done with four objective values (two for hard conflicts and two for soft conflicts) for demonstration purposes the two objective values both for the hard and soft constraints are added with each other. Hence in the following figures a single value is presented for the hard and soft constraints.

In the hillclimbing process, a vector is utilized which keeps the conflicts for the courses. When a hillclimber operator is applied, courses are selected either randomly or from this conflict vector. A parameter named *hcrand* is used to determine if the courses will be selected randomly or based on their conflict number. Higher randomness yields to better results. When randomness is decreased by this parameter, it has been observed that premature convergence becomes more probable based on the guidance provided by the conflict vector.

There are two parameters to control the amount of hill climbing that will be utilized. *Hcsize* defines the percentage of the pareto front that will go through hill climbing and *hcovsize* determines the hillclimbing percentage for the rest of the population. Due to computational cost, low values of *hcovsize* are preferred. Most of the time hill climbing is applied to the pareto front. The actual value for *hcsize* in the experiments is 100% and is *hcovsize* 10%.

Number of iterations used in group mutations is another parameter named as *hciter*. Increasing the number of iterations improves the local search but it results in a considerable amount of computational cost. Hence the parameter is set as 8 in the experiments.

The effect of applying the hill climber named as hill-climb-single with a value of 0.04 can be seen in Figures 6.4 and 6.5. While fitness values for hard conflicts are

improved, soft ones become worse. Since the hill climber is designed for hard conflicts the result is expected.

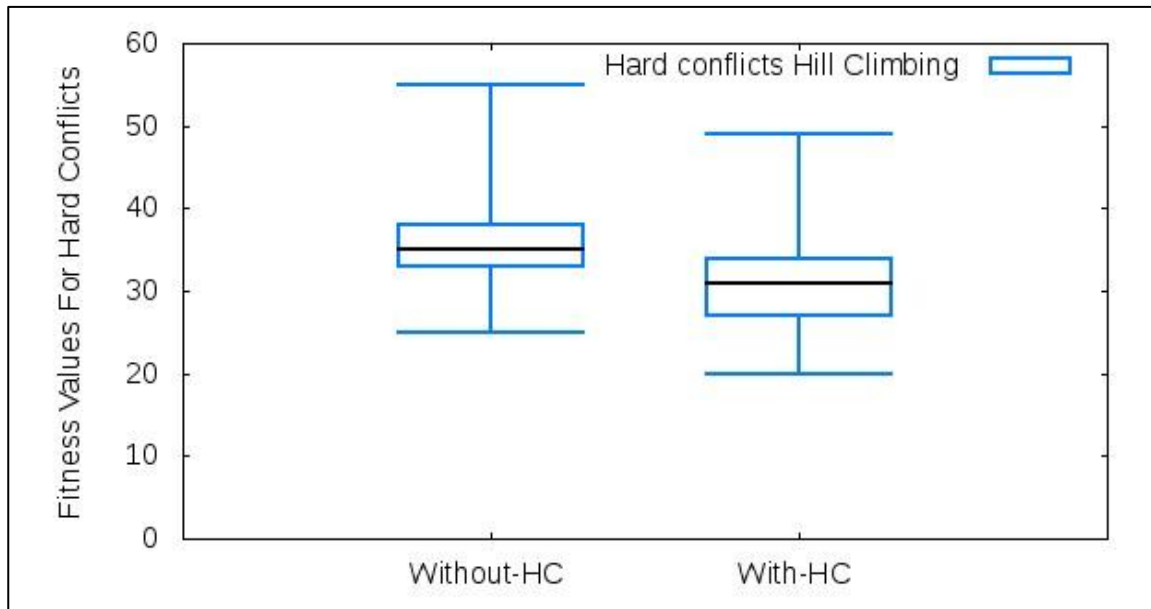


Figure 6.4. The effect of applying hill climbers

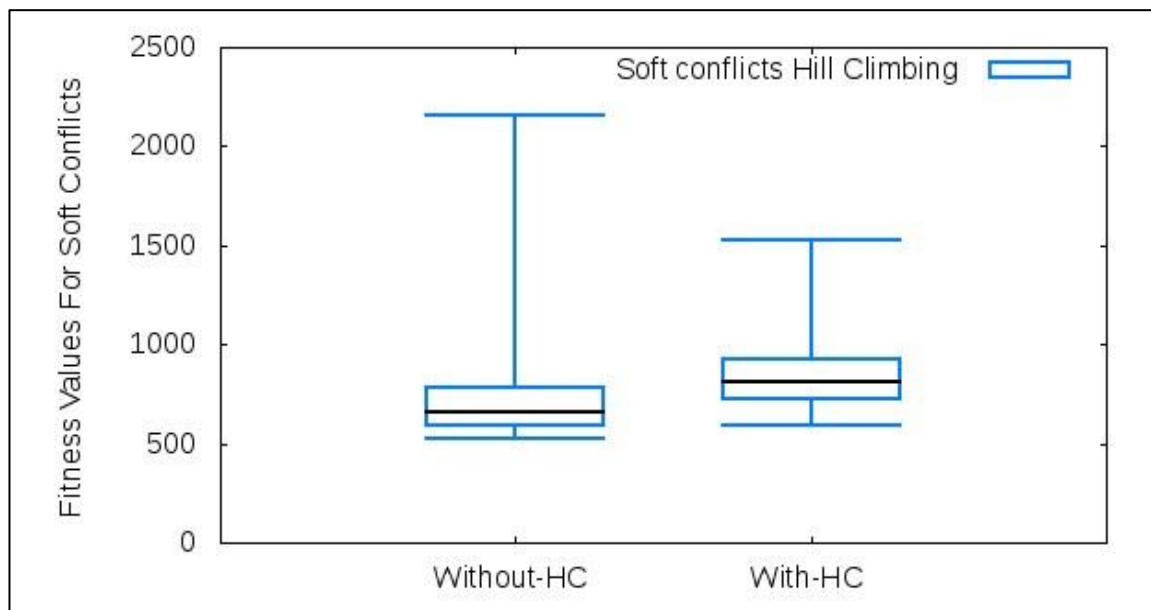


Figure 6.5. The effect of applying hill climbers

In order to see the effect of newly introduced crossover and selection operators the following schemes are used. These are single point crossover with binary tournament selection SX-BTS, uniform crossover with binary tournament selection UX-BTS, uniform crossover with crowding distance selection UX-CDS and group crossover with binary tournament selection GX-BTS.

These tests are done on Intel Core i5 2.27 GHz CPU with 4 GB RAM. Results are gathered over 20 runs each one using population size of 40. The duration of each run is 300 seconds. These tests are applied to *comp18*(small),*comp03* (medium) and *comp09*(large) instances of the benchmark problems of UCT-UU.

In Table 6.3, UX-BTS has minimum average fitness value for the hard conflicts for all datasets. The variance of UX-BTS is also minimum at the large instance. UX-CDS has minimum variances for small and medium datasets. In Table 6.4, UX-BTS has minimum average fitness value for soft conflicts for the medium and the large datasets. In the same table UX-CDS has best performance on small dataset and has minimum variances in all datasets.

Table 6.3. Average fitness values and deviations from the average for different crossover-selection pairs for hard conflicts.

Crossover- Selection	Average fitness	Standard deviation	Average fitness	Standard deviation	Average fitness	Standard deviation
	comp18	comp18	comp03	comp03	comp09	comp09
SX-BTS	14.878	2.886	71.081	5.345	79.055	5.379
UX-BTS	10.974	2.375	56.139	6.396	59.056	5.144
UX-CDS	11.090	2.128	63.661	5.327	67.639	5.477
GX-BTS	13.583	3.512	62.789	6.068	63.996	8.199

Table 6.4. Average fitness values and deviations from the average for different crossover-selection pairs for soft conflicts.

Crossover-Selection	Average fitness	Standart deviation	Average fitness	Standart deviation	Average fitness	Standart deviation
	comp18	comp18	comp03	comp03	comp09	comp09
SX-BTS	264.219	26.571	793.495	186.087	963.778	168.749
UX-BTS	253.636	36.463	647.854	117.765	685.594	141.675
UX-CDS	251.520	24.643	666.523	88.234	692.597	102.600
GX-BTS	277.720	50.446	657.751	114.952	692.558	111.718

Two way ANOVA with Tukey HSD post hoc tests for multiple comparisons are done for statistical analysis. These test results can be found in Figures 6.6 through 6.11. As seen in Figure 6.6, both UX-BTS and UX-CDS are statistically significant than SX-BTS and GX-BTS. In Figure 6.7 there is no significant difference between the algorithms.

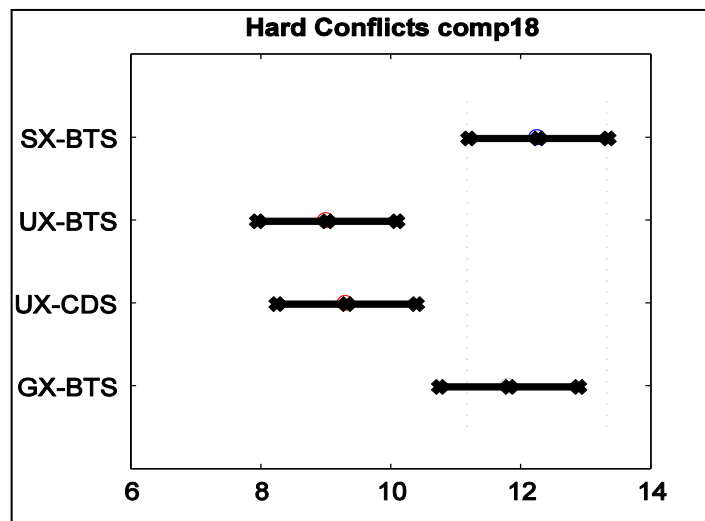


Figure 6.6. Comparison of selection crossover pairs for instance comp18.ctt

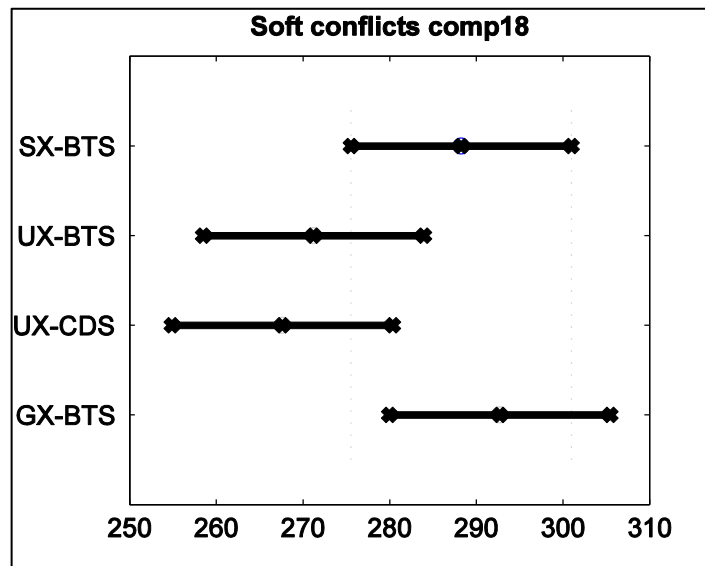


Figure 6.7. Comparison of selection crossover pairs for instance comp18.ctt

In Figure 6.8, the performance of UX-BTS, UX-CDS and GX-BTS are statistically significant than SX-BTS. UX-BTS has the best performance. In Figure 6.9 UX-BTS, UX-CDS and GX-BTS perform better than SX-BTS since this scheme seems to have a better performance compared to the other schemes utilized.

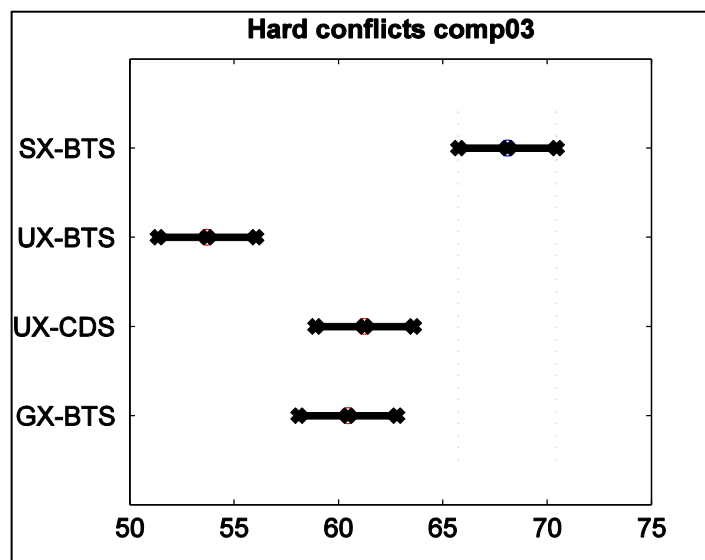


Figure 6.8. Comparison of selection crossover pairs for instance comp03.ctt

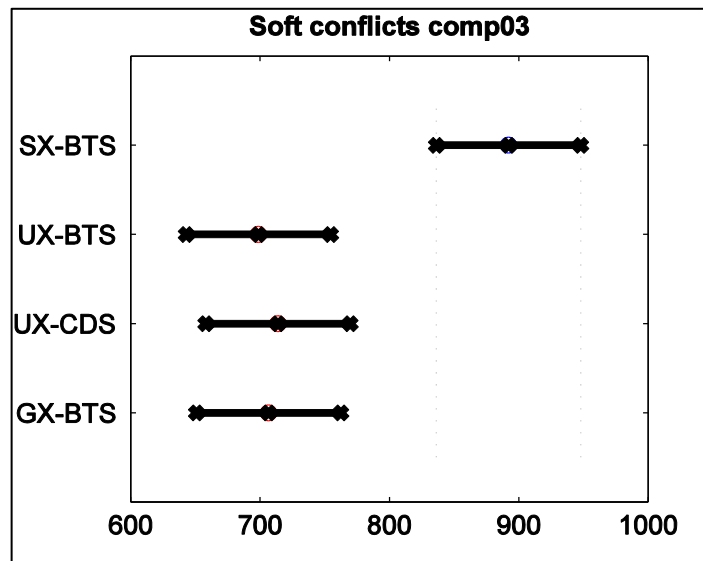


Figure 6.9. Comparison of selection crossover pairs for instance comp03.ctt

In Figure 6.10, UX-BTS, UX-CDS and GX-BTS are statistically significant than SX-BTS and UX-BTS performs better than UX-CDS. In Figure 6.11 UX-BTS, UX-CDS and GX-BTS perform better than SX-BTS. As a result UX-BTS selection-crossover scheme is used for the rest of the tests.

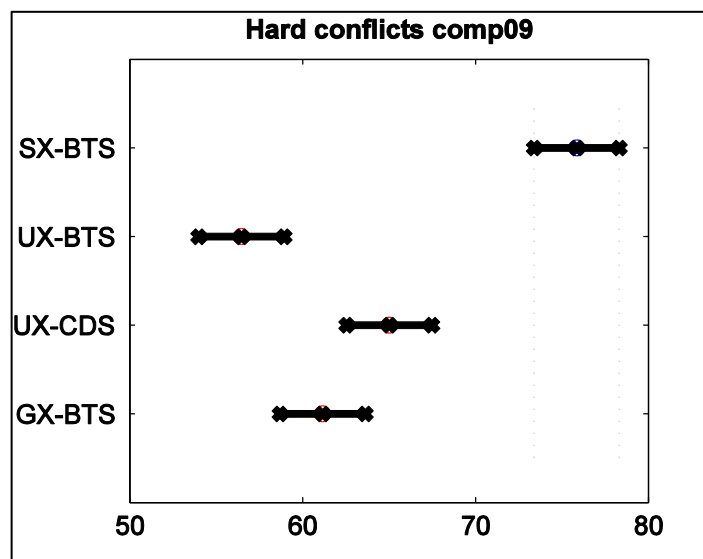


Figure 6.10. Comparison of selection crossover pairs for instance comp09.ctt

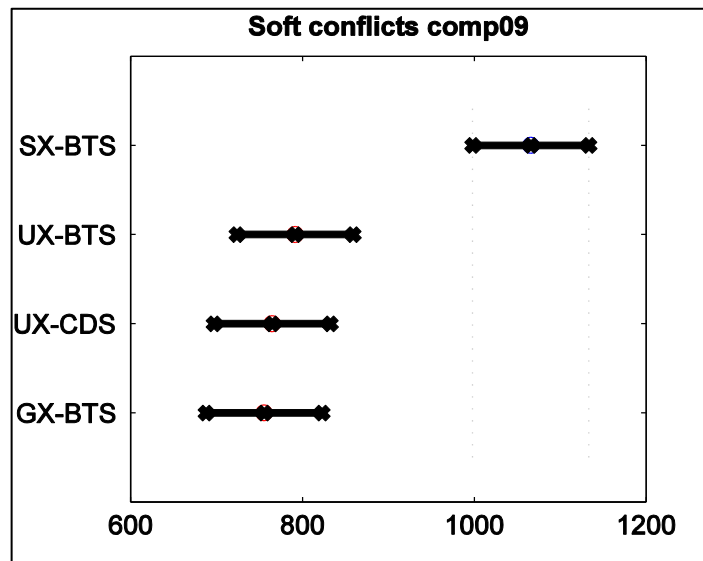


Figure 6.11. Comparison of selection crossover pairs for instance comp09.ctt

In UCT-UU benchmarks there are 20 instances to be tested which are very diverse in their sizes. 10 tests are done for each instance and the best result is recorded. Some instances and the results are shown in Table 6.6. These results are the best results over 10 experiments using different seeds. The results of MOGA used in this study is compared with the Top 5 results obtained in the literature [26]. Top 5 results use one of the following algorithms: Iterated Tabu Search, Threshold accepted local search and Repair based local search. These algorithms are local search based state of the art algorithms which are designed for competition purposes. They are all developed to solve directly the benchmark instances of UCT-UU. Our algorithm is achieved to find feasible timetables in all eight instances. In two of the instances, when quality of timetable is considered (i.e. soft constraints) the performance of MOGA is among the Top 5 algorithms in the literature.



Table 6.6. Benchmark results

Rank/Instance	Alg_1	Alg_2	Alg_3	Alg_4	Alg_5	MOGA hard-soft conf.
Comp01	0,5	0,5	0,5	0,5	0,10	0,17
Comp03	0,84	0,71	0,82	0,128	0,119	0,244
Comp05	0,330	0,309	0,312	0,410	0,426	0,1072
Comp09	0,109	0,105	0,110	0,150	0,139	0,204
Comp11	0,0	0,0	0,0	0,0	0,3	0,0
Comp12	0,333	0,343	0,351	0,442	0,408	0,892
Comp14	0,59	0,57	0,59	0,90	0,84	0,147
Comp18	0,83	0,69	0,68	0,116	0,110	0,114

## 7. CONCLUSION

When we consider the UCT-UU case, Top 5 algorithms are local search based state of the art algorithms which are designed directly for the benchmark instances of UCT-UU. Our algorithm is achieved to find feasible timetables in all eight instances. For two of the instances, the performance of MOGA is compatible with the top five state of the art algorithms in the literature. The results obtained are promising and can be improved further with additional mutational and hill climbing operators as future work.

In UCT-YU case our algorithm is able to find feasible and decent timetables for Yeditepe University Computer Science Engineering Department. The automated timetable created by MOGA is far better than the manual one. Since fall-2009 semester the timetables used in the department are created using MOGA.

The contributions of this thesis can be summarized as follows. Basically, this study proposes a multi objective GA to solve the timetabling problem as a graph coloring problem. Most of the approaches in the literature used two stage algorithms. First, feasible timetables are found and then another algorithm tries to improve the quality in terms of soft constraints. Our approach solves both hard and soft objectives simultaneously. We believed and insisted on Genetic Algorithms based on the generality of the methodology. Since domain specific knowledge is minimum in this framework, the proposed algorithm can be well applied to other similar problems (like scheduling) with minor updates. Multi objectivity provided a set of solutions instead of a single one. Increasing the multi objectivity level yields better timetables. Yeditepe University CSE department has some special constraints that cannot be found in the literature. These constraints are also handled successfully in this framework.

## REFERENCES

1. S.N. Sivanandam, S. N. Deepa, *Introduction to Genetic Algorithms*, Springer, 2010.
2. A. E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*, Springer, 2003.
3. Weise, T., *Global Optimization Algorithms Theory and Application*, E-book, 2007.
4. Deb, K., S.Agrawal, A.Pratap and T.Meyarivan “A Fast Elitist Non-dominated Sorting Genetic Algorithm for Multi-objective Optimization: NSGA-II”, in *Parallel Problem Solving from Nature PPSN VI, vol. 1917 Lecture Notes in Computer Science*, pp. 849--858, Springer, Berlin / Heidelberg, 2000.
5. Silva, A., E.K. Burke and S.Petrovic, “An introduction to Multiobjective Metaheuristics for Scheduling and Timetabling”, *Metaheuristic for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems*, 91--129, Springer, 2004.
6. Korkmaz, E., “Multi-objective Genetic Algorithms for Grouping Problems”, *Applied Intelligence*, Vol. 33, No. 2, pp. 179-192, 2008.
7. Alkan, A., “Memetic Algorithms for Timetabling”, in *Proc. of IEEE Congress on Evolutionary Computation* pp. 1796--1802, 2003.
8. Daskalaki, S., T.Birbas and E., “An Integer Programming Formulation For a Case Study in University Timetabling”, in *European Journal of Operational Research*, Vol. 153, No.1, pp. 117 -- 135, 2004.
9. Abramson, D., “Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms”, 1991.

10. Beligiannis, G.N., C.N. Moschopoulos, G.P. Kaperonis and S.D. Likiothanassis, "Applying Evolutionary Computation to the School Timetabling Problem: The Greek Case", *Comput. Oper. Res.*, Vol.35, No.4, pp. 1265--1280, 2008.
11. Cheong, C., K.Tan and B.Veeravalli, "A Multi-objective Evolutionary Algorithm for Examination Timetabling", *Journal of Scheduling*, Vol.12, No.2, pp. 121--146, 2009.
12. Santiago-Mozos, R., S.Salcedo-Sanz, M.DePrado-Cumplido and C.Bousoño-Calzón, "A two-phase Heuristic Evolutionary Algorithm for Personalizing Course Timetables: A Case Study in a Spanish University", *Computers&Operations Research*, vol.32, no.7, pp. 1761 -- 1776, 2005.
13. Lewis, R. and B.Paechter, "Application of the Grouping Genetic Algorithm to University Course Timetabling", in *G. Raidl and J.Gottlieb (eds) Evolutionary Computation in Combinatorial Optimization*, pp. 144--153, Springer, 2005.
14. Salwani Abdullah, Edmund K. Burke and Barry McCollum "A Hybrid Evolutionary Approach to the University Course Timetabling Problem.", *Journal of the Operational Research Society*, Vol. 58, No. 11, pp. 1494-1502, 2007.
15. Socha K., J. Knowles , M. Sampels, "A MAX-MIN Ant System for the University Course Timetabling Problem ", in *Proceedings of the 3rd International Workshop on Ant Algorithm*, pp:1—13, Springer, 2002.
16. Burke E. K., G. Kendall, E. Soubeiga, "A Tabu-Search Hyperheuristic for Timetabling and Rostering", *Journal of Heuristics*, Vol. 9, No. 6, pp: 451—470, 2003.
17. Azevedo F., P. Barahona "Timetabling in Constraint Logic Programming", in *World Congress on Expert System*, 1994.
18. Ülker, Ö., E.Özcan and E.Korkmaz, "Linear Linkage Encoding in Grouping Problems: Applications on Graph Coloring and Timetabling", in *Practice and Theory of Automated*

*Timetabling VI*, vol. 3867 of *Lecture Notes in Computer Science*, pp. 347--363, Springer, Berlin, 2007.

19. Blochliger I., N. Zufferey, “A graph coloring heuristic using partial solutions and a reactive tabu scheme”, *Computers & Operations Research*, Vol. 35, No. 3, pp 960-975, 2008.

20. Malaguti E., M. Monaci, and P. Toth. “A Metaheuristic Approach for the Vertex Coloring Problem”. *Inform Journal on Computing*, Vol. 20, No 2, pp. 302, 2008.

21. Porumbel D., J.K. Hao and P Kuntz, “A Search Space "Cartography" for Guiding Graph Coloring Heuristics”. *Computers & Operations Research*, Vol. 37, No. 4, pp. 769-778, 2010.

22. Hertz A., M. Plumettaz, and N. Zufferey. “Variable Space Search for Graph Coloring”. *Discrete Applied Mathematics* , 156(13): 2551-2560, 2008.

23. Michalewicz Z., *Genetic Algorithms + Data Structures=Evolution*, Springer Berlin, 1996.

24. Horn, J., N.Nafpliotis and D.E.Goldberg, “A Niche Pareto Genetic Algorithm for Multiobjective”, in *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pp. 82--87, 1994.

25. Fonseca, C.M. and P.J. Fleming, “Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and”, *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 416—423, 1993.

26. McCollum B., A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu and E. K. Burke, “Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition”, 2008.