Int. J. Advance. Soft Comput. Appl., Vol. 2, No. 1, March 2010
ISSN 2074-8523; Copyright © ICSRS Publication, 2010
www.i-csrs.org

Fuzzy Genetic Heuristic for University Course Timetable Problem

Arindam Chaudhuri[1] and Kajal De[2]

[1]Lecturer (Computer Science Engineering),
Birla Institute of Technology Mesra,
Patna Campus, Patna, India
e-mail: arindam_chau @ yahoo.co.in

[2]Professor of Mathematics, School of Science,
Netaji Subhas Open University, Kolkata, India

Abstract

University Course Timetable Problem is NP-Hard combinatorial optimization problem which lacks analytical solution methods. It has received tremendous attention from disciplines like Operations Research and Artificial Intelligence during past few years given its wide use in universities. Several algorithms have been proposed most of which are based on heuristics like Search techniques and Evolutionary Computation. We present Fuzzy Genetic Heuristic Algorithm to solve the problem. The method incorporates Genetic Algorithms using indirect representation based on event priorities, Micro Genetic Algorithms and heuristic Local Search operators to tackle real world Timetable Problem from St. Xavier's College, India. Fuzzy Set models measure of violation of soft constraint in fitness function to take care of inherent uncertainty and vagueness involved in real life data. The solutions are developed with respect to manual solution developed by College staff. The proposed technique satisfies all hard constraints of problem and achieves significantly better score in satisfying soft constraints. The algorithm is computationally intensive in comparison to standard Genetic Algorithm based benchmark heuristics. The reduction computational complexity of the algorithm can be considered as future work for further research.

Keywords: *Fuzzy Genetic Heuristic*, *Hard Constraints*, *Soft Constraints*, *University Course Timetable Problem*

1 Introduction

University Course Timetable Problem (UCTP) [7], [17], [51], [65] represents an important class of optimization problem in Operations Research. It is considered as one of the most difficult problems faced by universities and colleges today. The problem can be defined as allocation of given resources (teachers, students and classrooms) to objects (courses) being placed in space time satisfying all university constraints and optimizing utilization of existing facilities such that a set of desirable objectives are satisfied. Basically, university timetable problem exists in two forms viz., course and exam timetable formats. Here our focus is only on course timetable problem. The university course timetable requires several slots and with different categories such as lectures, tutorials and practical sessions, which fits within a week and repeats for whole semester. Given the increasing number of students in universities, a large number of courses are offered every term. Each course has different number of enrolled students and each classroom has different capacities which make assignment of courses to classrooms complicated. Furthermore, it is not only enough to schedule course in classroom with higher capacity than the number of enrolled students, since this can still lead to inefficient utilization of classrooms which can cause difficulties for teachers and students. The automation of timetable problem is thus an important task as it saves lot of man-hours work to institutions and provides optimal solutions with constraint satisfaction that can boost productivity, quality of education and services. However, large-scale timetables such as university timetables may need many hours of work spent by qualified person or team in order to produce high quality timetables with optimal constraint satisfaction [37] and optimization of timetable's objectives at the same time.

The real life timetable problems have many forms like education timetable (course and exam), employee timetable, timetable of sports events, timetable of transportation etc. Timetable problems as well as scheduling problems are generally NP-Hard constrained optimization problems [13], [37] of combinatorial nature and no optimal algorithm is known which generates solution within reasonable time. These problems are mainly classified as constraint satisfaction problems [5]. There are number of versions of UCTP differing from one university to another [18], [59]. A lot of work has been done on this type of problem with respect to their studies on specific universities and many formulations and algorithms have been developed One of the important computing paradigms is graph coloring concept [31], [53] where vertices represent courses and an arc joins two vertices only if they cannot be scheduled at same time. The problem is thus to find chromatic number of resulting graph [11]. However, chromatic number problem is also NP-Hard. Due to complexity of the problem, most of work done concentrates on heuristic algorithms which try to find good approximate solutions [1], [2], [3], [4], [6], [8], [9], [19], [21], [35], [36], [38], [39], [40], [41], [42], [43], [44], [49], [51], [56], [60], [62]. Some of these include Genetic Algorithms (GA) [14], [25], [48], [54], [57], [61], [62], Tabu Search [22], [30], [46], [64], Simulated Annealing [20], [55], [63] and recently used Scatter Search [45] methods. Heuristic optimization methods are explicitly aimed at good feasible solutions that may not be optimal where complexity of problem or limited time available does not allow exact solution. Generally, two questions arise (i) How fast

the solution is computed? and (ii) How close the solution is to optimal one? Tradeoff is often required between time and quality which is taken care of by running simpler algorithm more than once, comparing results obtained with more complicated ones and effectiveness in comparing different heuristics. The empirical evaluation of heuristic method is based on analytical difficulty involved in problem and pathological worst case result.

In recent past these heuristic tools have been combined among themselves with knowledge elements [1], [10], [15], [20], [25], [44], [55], [62], [63] as well as with more traditional approaches such as Statistical and Fuzzy Analysis [32], [33], [66] to solve extremely challenging problems. Developing solutions with these tools offers two major advantages viz. (i) Shorter development time than traditional approaches and (ii) Robust systems being insensitive to noisy and missing data. Keeping in view recent past, this work attempts to develop Fuzzy Genetic Heuristic (FGH) algorithm for university timetable at St. Xavier's College, Kolkata, India for which manual solutions are available. GA heuristic is combined with Fuzzy Sets to handle imprecisely defined parameters of problem which are apparent in real life data. Fuzzy Logic is a computational paradigm that generalizes classical two-valued logic for reasoning under uncertainty. In order to achieve this, notation of membership in a set needs to become a matter of degree. This is the essence of Fuzzy Sets [32], [66]. By doing this two things are accomplished (i) The ease of describing human knowledge involving vague concepts and (ii) The enhanced ability to develop cost-effective solution to real-world problem. It is multi-valued logic which is model-less approach and clever disguise of Probability Theory.

GA [12], [24] is search algorithm based on conjecture of natural selection and genetics. It is different from other search techniques in several aspects such as (i) The algorithm is multi-path that searches many peaks in parallel and hence reduces possibility of local minimum trapping; (ii) It works with coding of parameters instead of parameters themselves which help genetic operator to evolve current state into next state with minimum computations; (iii) The fitness of each string is calculated evaluated to guide its search instead of optimization function; (iv) There is no requirement for derivatives or other auxiliary knowledge such that no computation of derivatives or other auxiliary functions are required; (v) GA explores search space where probability of finding improved performance is high. Hence, GA is often viewed as black box approach. In contrast, Fuzzy Logic models are easy to comprehend because they use linguistic terms and structured rules. Unlike GA, Fuzzy Logic does not come with search algorithm. Fuzzy models adopt techniques from other areas such as Statistics, Linear System Identification etc. Since, GA has the search ability; it is natural to merge the two paradigms. This merger creates FGH algorithm, which describes Fuzzy Set model using GA search attribute. FGH uses an indirect representation featuring event allocation priorities and invokes timetable builder routine for constructing complete timetable. The algorithm incorporates number of techniques and domain specific heuristic local search operators to enhance search efficiency. The non-rigid soft constraints involved in the problem are basically optimization objectives for search algorithm, for which there is an

inherent degree of uncertainty involved in objectives which comprises of different aspects of real life data. This uncertainty is tackled by formulating measure of violation parameter of soft constraint in fitness function using fuzzy membership functions. The solutions are generated against manual problem of St. Xavier's College and compared with respect to standard GA based benchmark problems. It has been shown through extensive simulation that on incorporating certain combinatorial and domain specific operators search efficiency of GA is significantly enhanced. To performance of GA is further improved through Micro GA which explores the use of small population size. This Paper is organized as follows. The section 2 illustrates UCTP. This is followed by discussion of various uncertainty measures involved in UCTP. The next section presents FGH algorithm for UCTP. The simulation results are given in section 5. The section 6 gives the conclusions. Finally, future work underlying the problem is given in section 7.

2 University Course Timetable Problems

UCTP [17], [40], [42], [52], [65] consists in finding the exact time allocation within limited time period of number of events (courses-lectures) and assign to them number of resources (teachers, students and classrooms) such that the constraints are satisfied. In most universities courses are organized in number of semesters. The constraints to be satisfied by timetable are usually divided into two categories viz. *hard* and *soft* constraints. Hard constraints should be rigidly fulfilled. Such constraints include: (i) No resource (teachers, students and classrooms) may be assigned to different events at same time; (ii) Events of same semester must not be assigned at same time slot (in order for students of semester to be able to attend all semester courses); (iii) Assigned resources to an event must belong to set of valid resources for that event. In this regard, the lecture is held in a classroom if proper infrastructural arrangements are there to organize the lecture. Similarly, the lecture is assigned to teacher if he has knowledge as well as capability of delivering particular lecture. Likewise, some lectures are assigned to teachers if they have knowledge to deliver the lecture. On the other hand, it is desirable to fulfill soft constraints to the possible extent but is not fully essential for valid solution. Therefore, soft constraints can also be seen as optimization objectives for search algorithm. Such constraints are: (i) Schedule an event within particular window of whole period (such as during evenings); (ii) Minimize time gaps or travel times between adjacent lectures of same teacher, etc. The problem considered for this work is taken from St. Xavier's College, Kolkata, India and involves weekly scheduling of all courses of Department of Computer Science. The problem specifications are given in Table 1. Hard and soft constraints considered for this problem are given in Table 2 and 3 respectively.

In Table 1, value 10 for the field time-periods within a day denote possible starting periods of each class (from 8:00 am to 6:00 pm) and not complete time slots that can accommodate equal number of consequent classes. As different lectures have different durations (1 to 2 hours), real number of consequent classes that can be scheduled within a day depends on specific set of classes chosen and their durations. Any solution satisfying

above constraints is feasible schedule for the problem. The specific case is considered as benchmark and reasons are: (i) The real constraints were easily accessed for developing manual solution to problem, in order to set-up university course timetable problem on realistic basis; (ii) There was an easy access to manual solutions for the problem which facilitates making easy comparisons with present results; (iii) The specific problem is generally a NP-Hard problem and serves as demanding benchmark for developing an efficient optimization algorithm. There are certain difficulties involved in chosen problem case which are justified by following facts: (i) Problem has two types of lectures viz. *theory* and *laboratory* with diverse characteristics and constraints; (ii) Number of classrooms is generally small (viz. only 19) in College that accommodates all taught lessons, a fact which makes timetable schedule very tight. Some classrooms are laboratories designed for *laboratory* classes and others are *theory* classrooms. All laboratories are occupied by classes for full number of periods per day and all five days with only minor time-gaps; (iii) Specific classes are taught in specific classrooms. *Theory* classes are assigned to any of the lecture classrooms, but *laboratory* classes must be assigned to specific laboratory classrooms; (iv) There are quite large number of teachers, each of whom has their own minimum (4) and maximum (12) hour limits per week and ability to teach in limited set of classes.

Table 1: Timetable Problem Specifications

Serial Number	Parameter Description	Quantity
1	Number of Courses	90
2	Number of different Lectures	200
3	Number of scheduled Events	210
4	Number of Semesters	11
5	Type of Lectures (Theory/Laboratory)	2
6	Number of Teachers	50
7	Number of Classrooms/Laboratories	19
8	Number of Days	5
9	Number of Periods within a Day	10

3 Uncertainty Measures in University Course Timetable Problem

In this section, we discuss various uncertainty measures involved in formulating UCTP. The uncertainty measures are associated with soft constraints of problem. Fuzzy Sets are used to model uncertainty and vagueness associated with soft constraints in final timetable schedule by allowing grades of membership in the set. The model allows decision maker to express his preference to ultimate schedule such that related measure of violation is appropriately represented. Among soft constraints, best availability schedule of each teacher, maximum and minimum workload of each teacher as well as classes broken into more than one non-contiguous lecture within a week where specific number of days are left between lectures are uncertain due to both human as well as environment factors. In addition, travel time of teachers and students between rooms

within campus, time gaps within schedule of each teacher and time gaps within schedule of each room which have to be minimized, have an inherent degree of uncertainty and impreciseness factors associated. These constraints are now represented using Fuzzy Sets.

Table 2: Hard Constraint Specifications

Serial Number	Hard Constraint
1	No resource (teacher, student or classroom) is assigned to different events at same time
2	Events of same semester are not assigned at same time slot when both events are of type *theory* or when one event is *theory* and other event is *laboratory*. Same semester events run concurrently only if they are both of type *laboratory*, as for each course 4 *laboratory* classes are scheduled within a week, each attended by different group of students.
3	Maximum number of time periods per day should not exceed particular value (10)
4	Each lecture is held in a classroom belonging to specific set of valid rooms for lecture
5	Each classroom has its own availability schedule
6	Each lecture is assigned to a teacher that belongs to specific set of teachers that can deliver lecture
7	Specific lectures must be rigidly assigned to specific teachers.
8	*Theory* classes need one teacher while *Laboratory* classes need two teachers

Table 3: Soft Constraint Specifications

Serial Number	Soft Constraint
1	Every teacher has his own availability schedule ensuring which he submits plan with desirable time periods that suits him best
2	Every teacher has minimum and maximum limit of weekly work-hours which are 12 and 4 respectively
3	If class is broken in more than one non-contiguous lectures within a week, specific number of days must be left between these lectures
4	Travel time of teachers and students between classrooms within campus is to be minimized
5	Time gaps within schedule of each teacher is to be minimized
6	Time gaps within schedule of each classroom is to be minimized

Definition: A fuzzy set $\widetilde{A}$ is defined by membership function $\mu_{\widetilde{A}}(x)$ which assigns to each object *x* in universe of discourse *X*, a value representing its grade of membership in fuzzy set given by [32], [66],

$$\mu_{\tilde{A}}(x) : X \rightarrow [0,1] \qquad (1)$$

A variety of shapes are used to represent fuzzy memberships such as triangular, trapezoidal, bell-curves, s-curves etc. Conventionally, choice of curve shape is subjective and allows decision maker to express his preferences. The estimation of time elapsed with respect to soft constraints 4th, 5th and 6th is obtained by taking into consideration nature of teacher, student and location of rooms. While some people walk faster, others may walk slowly as a result of which elapsed time are basically dependent on walking speed of different people. Uncertain elapsed times $\tilde{p}_{ij}$ are modeled by using triangular membership functions [32], [66] represented by triplet$(p_{ij}^1, p_{ij}^2, p_{ij}^3)$, where p_{ij}^1 and p_{ij}^3 are lower and upper bounds of elapsed time while p_{ij}^2 is modal point as represented in Fig.1. The use of triangular fuzzy numbers to model uncertainty in elapsed times may be attributed to the fact that three state representations through triplet$(p_{ij}^1, p_{ij}^2, p_{ij}^3)$ most accurately simulate real life data available.

Fig. 1: Fuzzy representation of Elapsed Times

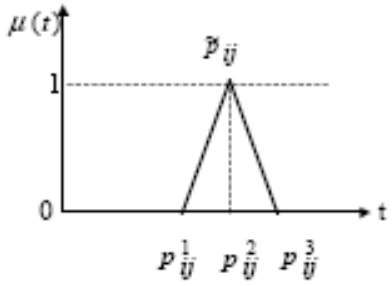
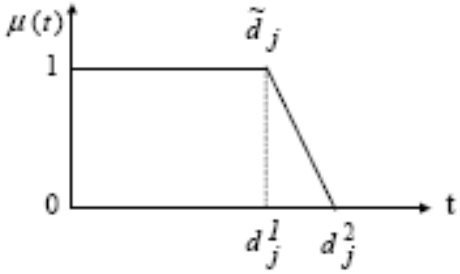


Fig. 2: Fuzzy representation of Schedule of Teacher



The 2nd soft constraint i.e. weekly work-hours of each teacher can similarly be represented by triangular membership functions [32], [66] represented by triplet$(w_{ij}^1, w_{ij}^2, w_{ij}^3)$, where w_{ij}^1 and w_{ij}^3 are minimum and maximum bounds of weekly work-hours of each teacher while w_{ij}^2 is modal point. The similar justification for using triangular fuzzy membership functions hold for weekly work-hours also.

The soft constraints 3rd and 1st are represented using LR trapezoidal membership functions [32], [66]. The specific number of days must be left between non-contiguous

lectures within a week is given by days elapsed $\widetilde{d}_j$ and represented by doublet (d_j^1, d_j^2), where d_j^1 and d_j^2 denote left and right end of trapezoid as depicted in Fig. 2. The schedule of each teacher with respect to his own availability and desirable time periods that suits him best is given by schedule $\widetilde{s}_j$ and represented by doublet (s_j^1, s_j^2), where s_j^1 and s_j^2 denote left and right end of trapezoid. Likewise, use of left and right end of trapezoid effectively models real life data available for 3rd and 1st constraints.

4 Fuzzy Genetic Heuristic for University Course Timetable Problem

This section presents FGH algorithm for UCTP. To solve the timetable problem, we develop an optimization method based on FGH that incorporates number of techniques and domain specific local search operators. GA an iterative search procedure widely used in solving optimization problems, motivated by biological models of evolution. A population of candidate solution is maintained in each iteration [12], [24]. Genetic operators such as mutation and crossover are applied to evolve solutions and find good solution that has high probability to survive for next iteration. First, representation method is required to encode timetable solution into an encoded form or chromosome suitable for applying genetic operators. Generally, two different approaches are considered viz. *direct* and *indirect* approaches. A *direct* representation [2] directly encodes all event attributes viz. day, time slot, teacher, classroom etc. for all events. In these cases GA has to decide for all timetable parameters and deliver complete and constraint free schedule. This results in very large search space having solutions satisfying all constraints. However, directly encoded solutions, that undergo genetic operators, frequently result in invalid solutions that have to be handled in some manner. An *indirect* representation [49] on other hand, considers encoded solution i.e. chromosome that usually represents an ordered list of events which are placed into timetable according to some predefined method (*timetable builder*). The *timetable builder* can use any combination of heuristics and local search to place events into timetable while observing constraints of the problem.

For GA implementation of this work, we have considered an indirect representation that encodes four fields for each event into chromosome: (i) Day to allocate event; (ii) Teachers (1 or 2) to assign to event; (iii) Classroom where event will be held; (iv) Priority to allocate event within day. All fields are first encoded as integers and then entered into chromosome as binary numbers. When GA produces such a solution, it first decodes it to gain these four fields for every event in schedule. Then it invokes *timetable builder* routine viz. *timetabler* that works as follows: (i) It separates events into clusters, one for each day; (ii) For every cluster, it sorts events according to their *priority* values and in ascending order (small values are with high priority and are placed first); (iii) It takes first event in cluster (the event with higher *priority*), marks it as taken, and places it into schedule of particular day; (iv) Starting from time slot 1 it places event and checks if any constraints are violated. If allocation is not fixed, algorithm moves on to next event in

cluster; (v) If any constraints are violated, they are allocated to event into subsequent time periods until all constraints are satisfied; (vi) If there exists no time period for which all constraints are satisfied, event is marked to violate *maximum time periods exceeded per day* constraint (3rd constraint from Table 2); (vii) The algorithm continues with next event in list. When all events have been processed, *timetabler* moves to next cluster (day), and this is repeated for all days in the schedule. A similar algorithm has been given in by [6] where non-evolutionary heuristic algorithm is proposed for examination timetable problem. All events are sorted according to *measure of difficulty* figure that is dynamically adapted during run and difficult to schedule events are handled first. In this work, allocation priority of events is determined genetically. The *timetabler* satisfies all hard constraints of Table 2 and all other constraints are satisfied by GA.

4.1 Formulation of Fitness Function

Now we discuss the formulation of fitness function for FGH algorithm. After *timetabler* has produced timetable, it is evaluated through fitness function that analyzes the solution and calculates its overall fitness values as sum of weighted scores and penalties for all constraints i.e., hard and soft. The fitness function used here is [12], [24], [32], [33], [66],

$$F(x) = \sum_{i=1}^{6} w_i^s \times P_i^{soft}(x) + \sum_{i=1}^{8} w_i^k \times P_i^{hard}(x) \qquad \textbf{(2)}$$

where, x is timetable under evaluation, $P_i^{soft}(x)$ is measure of violation of i^{th} soft constraint, $P_i^{hard}(x)$ is measure of violation of i^{th} hard constraint, w_i^s is weight factor for i^{th} soft constraint and w_i^h is weight factor for i^{th} hard constraint. The weights w_i^s and $w_i^h \in$ [0, 1] used in GA are normalized and randomly distributed, as weights are not known with certainty. In order to explore different areas of search space, in every iteration weights are changed. The function $F(x)$ is to be minimized. The impreciseness and uncertainty aspects related to measure of violation of soft constraints are taken care of using Fuzzy Logic. However, impreciseness and uncertainty aspects related to measure of violation of hard constraints are handled using probabilistic measures as constraints are defined rigidly. The nature of fitness function is dependent on both measures of violation $P_i^{soft}(x)$ and $P_i^{hard}(x)$ which assesses ultimate quality of different allocations within population. Fitness function value is changed in every iteration of algorithm, in order to explore different areas of the search space. Since, changes in weights affects final solution, these changes are basically random in nature.

One of measures of violation for soft constraints is calculated taking into consideration estimation of time elapsed with respect to different resources and events. Fuzzy elapsed times between resources and events imply fuzzy completion times. The question arises how to compare fuzzy completion times with fuzzy elapsed times between resources and events. This is investigated here based on possibility measure [19]. Possibility measure $\pi_{\widetilde{V}_j}(\widetilde{p}_{ij})$ evaluates possibility of fuzzy event, $\widetilde{V}_j$ occurring within fuzzy set $\widetilde{p}_{ij}$. It is used to compute measure of violation of time elapsed with respect to different resources and

events such that fuzzy completion times are minimized. Since, uncertainty in elapsed times $\widetilde{p}_{ij}$ are modeled by using triplet $(p_{ij}^1, p_{ij}^2, p_{ij}^3)$, we consider possibility measure as being composed of two fuzzy events $\widetilde{V}_{1j}$ and $\widetilde{V}_{2j}$. The event $\widetilde{V}_{1j}$ is considered for pair (p_{ij}^1, p_{ij}^2) and event $\widetilde{V}_{2j}$ considers pair (p_{ij}^2, p_{ij}^3) of elapsed times. Thus, measure of violation is given by,

$$P_{time-elapsed}^{soft}(x) = \pi_{\widetilde{V}_j}(\widetilde{p}_{ij}) = 1 - \sup\min\{\sup\min\{\mu_{\widetilde{V}_{1j}}(x), \mu_{\widetilde{p}_{ij}}(x)\}, \sup\min\{\mu_{\widetilde{V}_{2j}}(x), \mu_{\widetilde{p}_{ij}}(x)\}\} \quad (3)$$
$$j = 1,.....,n$$

where, $\mu_{\widetilde{V}_j}(x)$, $\mu_{\widetilde{V}_{1j}}(x)$, $\mu_{\widetilde{V}_{2j}}(x)$ and $\mu_{\widetilde{p}_{ij}}(x)$ are membership functions of fuzzy sets $\widetilde{V}_j$, $\widetilde{V}_{1j}$, $\widetilde{V}_{2j}$ and $\widetilde{p}_{ij}$ respectively.

The other measure of violation for soft constraints is calculated taking into consideration weekly work-hours of each teacher with respect to minimum and maximum bounds of weekly work-hours of each teacher. Fuzzy weekly work-hours of each teacher lead to computation of fuzzy maximum and minimum of weekly work-hours. The comparison of fuzzy weekly work-hours with fuzzy maximum and minimum of weekly work-hours of each teacher is performed using above possibility measure [19]. Possibility measure $\pi_{\widetilde{P}_j}(\widetilde{w}_{ij})$ evaluates possibility of fuzzy event, $\widetilde{P}_j$ occurring within fuzzy set $\widetilde{w}_{ij}$. It is used to compute measure of violation of weekly work-hours of each teacher with respect to minimum and maximum bounds of weekly work-hours of each teacher. As uncertainty involved in weekly work-hours $\widetilde{w}_j$ are modeled by using triplet $(w_{ij}^1, w_{ij}^2, w_{ij}^3)$, we consider possibility measure as being composed of two fuzzy events $\widetilde{P}_{1j}$ and $\widetilde{P}_{2j}$. The event $\widetilde{P}_{1j}$ is considered for pair (w_{ij}^1, w_{ij}^2) and event $\widetilde{P}_{2j}$ considers pair (w_{ij}^2, w_{ij}^3) of weekly work-hours. Thus, measure of violation is given by,

$$P_{work-hours}^{soft}(x) = \pi_{\widetilde{P}_j}(\widetilde{w}_{ij}) = 1 - \sup\min\{\sup\min\{\mu_{\widetilde{P}_{1j}}(x), \mu_{\widetilde{w}_{ij}}(x)\}, \sup\min\{\mu_{\widetilde{P}_{2j}}(x), \mu_{\widetilde{w}_{ij}}(x)\}\}; \quad (4)$$
$$j = 1,.....,n$$

where, $\mu_{\widetilde{P}_j}(x)$, $\mu_{\widetilde{P}_{1j}}(x)$, $\mu_{\widetilde{P}_{2j}}(x)$ and $\mu_{\widetilde{w}_{ij}}(x)$ are membership functions of fuzzy sets $\widetilde{P}_j$, $\widetilde{P}_{1j}$, $\widetilde{P}_{2j}$ and $\widetilde{w}_{ij}$ respectively.

The third measure of violation for soft constraints is calculated by considering schedule of each teacher with respect to his own availability and desirable time periods that suits him best. Fuzzy availability of each teacher gives fuzzy desirable time periods. The comparison of fuzzy availability with fuzzy desirable time periods is performed using possibility measure [19]. Possibility measure $\pi_{\widetilde{E}_j}(\widetilde{s}_j)$ evaluates possibility of fuzzy event, $\widetilde{E}_j$ occurring within fuzzy set $\widetilde{s}_j$. It is used to compute measure of violation of each teacher with respect to his own availability and desirable time periods that suits him best and is given by,

$$P^{soft}_{schedule}(x) = \pi_{\widetilde{E}_j}(\widetilde{s}_j) = 1 - \sup\min\{\mu_{\widetilde{E}_j}(x), \mu_{\widetilde{s}_j}(x)\};\ j = 1,.....,n \quad (5)$$

where, $\mu_{\widetilde{E}_j}(x)$ and $\mu_{\widetilde{s}_j}(x)$ are membership functions of fuzzy sets $\widetilde{E}_j$ and $\widetilde{s}_j$ respectively.

The fourth measure of violation for soft constraints is calculated by considering specific number of days left between non-contiguous lectures within a week and corresponding upper and lower bounds. Fuzzy specific number of days left between non-contiguous lectures within a week results in computation of fuzzy upper and lower bounds. The comparison between above two aspects is performed using possibility measure [19]. Possibility measure $\pi_{\widetilde{M}_j}(\widetilde{d}_j)$ evaluates possibility of fuzzy event, $\widetilde{M}_j$ occurring within fuzzy set $\widetilde{d}_j$. It is used to compute measure of violation of specific number of days left between non-contiguous lectures within a week and is given by,

$$P^{soft}_{days-left}(x) = \pi_{\widetilde{M}_j}(\widetilde{d}_j) = 1 - \sup\min\{\mu_{\widetilde{M}_j}(x), \mu_{\widetilde{d}_j}(x)\};\ j = 1,.....,n \quad (6)$$

where, $\mu_{\widetilde{M}_j}(x)$ and $\mu_{\widetilde{d}_j}(x)$ are membership functions of fuzzy sets $\widetilde{M}_j$ and $\widetilde{d}_j$ respectively.

From above discussion it is clear that, some of problem's constraints are handled by *timetabler* during construction of complete solution from genetically produced abstract solution. The rest of constraints are handled using penalty function that is composed as weighted sum of penalty terms, each of which corresponds to measure of violation of each constraint. Moreover, soft constraints could also be seen as optimization objectives that have to be optimized to possible extent.

4.2 Genetic Operators

The next issue to consider is the blend of genetic operators incorporated into GA, in order to achieve maximum optimization performance. To do this we first considered standard operators as well as general purpose combinatorial operators. The operators and their parameters considered are shown in Table 4. The standard GA setup employed Roulette Wheel Parent Selection, Population of 50 solutions, Standard 5-point Crossover Operator and Bit Mutation Operator (Probability = 0.001 per bit elitism) [12], [24]. The offspring replaced whole population of parents with Fitness scaling and generation limit of 7000 generations. The operators and their parameters of Table 4 were tested before adopting them in final algorithm. Due to specific nature and intractability aspect involved in problem we have also considered domain specific Hill Climbing Operators that are applied only to best solution of each generation. These operators include [12], [24]:

(i) **Change Day Hill Climbing Operator:** This operator selects an event at random (1st and 2nd constraints of Table 2) and changes its encoded *day of allocation* field, assigning to it all day values sequentially, except from original day value. Every time resulting timetable is evaluated, and if it scores better than original then change is kept otherwise old *day* value is restored.

(ii) **Fix Teacher Hill Climbing Operator:** This operator finds all events with *teacher class* constraint violations (6^{th} constraint of Table 2) and selects one such event at random. Then it changes encoded *teacher to allocate* field, assigning to it all valid teachers sequentially except from original one. Every time resulting timetable is evaluated, and if it evaluates better than original then change is kept otherwise old *teacher* value is restored. This operator is also successfully applied to 7^{th} and 8^{th} constraints of Table 2.

(iii) **Fix Classroom Hill Climbing Operator:** This operator considers events with *classroom availability* constraint violations (5^{th} constraint of Table 2) and selects one event at random. Then it changes encoded *classroom to allocate field*, assigning to it all valid classrooms sequentially except from original one. Every time resulting timetable is evaluated, and if it evaluates better than original then change is kept otherwise old *classroom* value is restored.

(iv) **Fix Room Hill Climbing Operator:** This operator finds all events with *classroom lecture* constraint violations (4^{th} constraint of Table 2) and selects one such event at random. Then it changes encoded *room to allocate* field, assigning to it all valid classrooms sequentially except original one. Every time resulting timetable is evaluated, and if it evaluates better than original then change is kept otherwise old *room* value is restored.

(v) **Fix Day Hill Climbing Operator:** This operator finds all events that are allocated beyond maximum *time periods per day* limit (3^{rd} constraint of Table 2), and selects one such event at random. Then it changes encoded *day of allocation* field, assigning to it all day values sequentially, except from original one. Every time resulting timetable is evaluated, and if it evaluates better than original then change is kept otherwise old *day* value is restored.

It is obvious that above operators are specifically designed to give GA ability to fulfill all hard constraints of Table 2. The effectiveness of these operators has been also tested and simulation results are given in the next section. The FGH Algorithm (pseudo-code) is illustrated in Fig. 3. In the FGH algorithm, the event priority is given by placing small values first with high priority. The algorithm is not swapping evenly between clusters as all the constraints are not satisfied always and there is violation between clusters. If an event is not scheduled then *timetabler* will generate inconsistent results.

5. Simulation Results

Simulations are performed for all parameter combinations of Table 4 before deciding best combination of operators to be adopted in ultimate implementation. However number of combinations is prohibitive for exhaustive evaluation. The different Genetic Operators are categorized as (a), (b), (c) etc. based on different values of parameters given in [27], [28], [29], [50]. Thus, *elitism like* technique has been applied in order to reduce number of simulations needed. First a simulation experiment was conducted for standard GA setup discussed in previous section. The experiment consisted of 20 independent runs. After completion of runs, three statistical figures were calculated viz. *the overall best solution quality achieved*, *the overall worst solution quality achieved* and *the average solution quality achieved* throughout 20 runs. Then, first operator setup of Table 4 was added to standard GA setup and another simulation round of 20 runs was launched. The results were compared to those of standard setup via three statistical figures mentioned above. If new setup had better performance than original, then new setup was adopted as *best so far* setup. Otherwise tested setup was ignored. With this method only 25 simulations of 20 runs each are needed to evaluate operators and their parameters of Table 4. The validity of this method is based on assumption that operators are more or less independent of each other, which is obviously true as justified by experimental results. The simulation results for operators of Table 4 are given in Table 5, where adopted setups are displayed in bold typeface. From Table 5 it is clear that operators that exhibited best performance and adopted in GA scheme are as follows [12], [24]:

(i) Uniform Crossover; (ii) Window Mutation Operator (Probability = 0.4); (iii) Swap Chromosome Operator (Probability = 0.1); (iv) Mutate Chromosome Operator (Probability = 0.1); (v) Varying Fitness Function with square increase; (vi) GA Population of 400 genotypes; (vii) Micro GA Combinatorial Hill Climbing Operator

Initialize Population of candidate solutions;
Compute Fitness function;
While (*stopping criterion is not satisfied*)
{
  Perform Mutation to evolve solutions;
  Perform Crossover to evolve solutions;
  Encode timetable solution into four fields;
  Decode to gain four fields for every event;
  Invoke timetabler:
        Separate events into *clusters* one for each day;
        For (*every cluster*)
        {
                Sort event according to their *priority* values in ascending order
                (*small* values are with high *priority* and are placed first);
                Take *first event* in *cluster* (event with higher *priority*), *mark* it and
                 place it into schedule of particular day;
                Starting from time slot 1,
                  Place event and *check* if any constraints are violated;
                  If (*allocation is not fixed*)
                     then move to next event in cluster;

```
                    If (any constraints are violated)
                       then allocate to event into subsequent time periods;
                    If (there exists no time period for which all constraints are satisfied)
                      then mark event to violate maximum time periods exceeded per
                           day constraint;
                    Continue with next event in list until all events are processed;
            }
            Repeat for all days in schedule;
    Evaluate timetabler through Fitness function that calculates overall fitness values;
    Update Population;
}
return results;
```

Fig. 3: Fuzzy Genetic Heuristic Algorithm (pseudo-code)

To improve the performance of GA in this work Micro GA [24] is used. The Micro GA strategy is derived by [24] explores the use of small population sizes on GA applications. Reeves [54] showed that for Binary encoding a small population size is sufficient to reach the entire search space by Crossover alone. With small populations used here there is a rapid convergence to possible sub-optimal solution and frequent regeneration of population members to ensure diversity during the search process. The Micro GA is effectively used is to repeatedly generate new population members as soon as a measure of convergence has been achieved in the cycle of GA operation. The Micro GA gives significant improvement in Fitness evaluation during the entire course of the optimization process. This reduces the overall computational effort thereby providing significant improvement in time complexity of the algorithm.

It may be mentioned that the population adopted here for GA implementation is quiet large. To achieve appreciable results micro setting of the population should be at least 40 to 50 percent of population considered.

Table 4: Standard Genetic Operators and Parameters considered

Serial Number	**Genetic Operator**	**Parameter**
1 (a)	Crossover	40-point
1 (b)	Crossover	Uniform
2 (a)	Mutation	Probability = 0.007
2 (b)	Mutation	Probability = 0.02
3 (a)	Window Mutation Operator	Probability = 0.1
3 (b)	Window Mutation Operator	Probability = 0.4
4 (a)	Swap Chromosome Operator	Probability = 0.1
4 (b)	Swap Chromosome Operator	Probability = 0.4
5 (a)	Swap Bit Operator	Probability = 0.1

5 (b)	Swap Bit Operator	Probability = 0.4
6 (a)	Swap Window Operator	Probability = 0.1
6 (b)	Swap Window Operator	Probability = 0.4
7 (a)	Random Genotype Operator	Probability = 0.1
7 (b)	Random Genotype Operator	Probability = 0.4
8 (a)	Mutate Chromosome Operator	Probability = 0.1
8 (b)	Mutate Chromosome Operator	Probability = 0.4
9	Bit Swap Mutate Hill Climbing Operator	Probability = 0.5
10	Window Swap Hill Climbing Operator	Probability = 0.5
11 (a)	Varying Fitness Function	Linear
11 (b)	Varying Fitness Function	Square
11 (c)	Varying Fitness Function	Exponential
12 (a)	Genetic Algorithm Population	200
12 (b)	Genetic Algorithm Population	400
13	Micro Genetic Algorithm Combinatorial Hill Climbing Operator	Probability = 1.0

By adding these operators to standard GA scheme we have managed to evolve best overall solution from value of 61087 for standard setup down to value of 22982 for advanced setup. The next step tests effectiveness of domain specific Hill Climbing Operators discussed in section 4. For this reason four more simulation experiments were conducted. Each experiment incorporated one of four domain specific operators. Again 20 runs were executed for each experiment and each time results were compared to *best so far* results. When an operator was found to enhance performance of GA optimizer it was adopted. The simulation results for these operators are shown in Table 6. As it is obvious from Table 6, each one of four domain specific operators enhance performance of GA optimizer and thus all four operators were adopted in final scheme. The domain specific operators managed to evolve best overall solution from value of 22982 for advanced setup down to value of 2809. The optimal solution for 2809 can be analyzed into two parts: (i) Hard constraints violation part which is 2000; (ii) Soft constraints violation part which is 809. The value 2000 for first part means that all hard constraints are fully satisfied at optimal solution. Similarly, value of 809 for second part means that all soft constraints are fully satisfied and that gaps within classrooms and teacher schedules are adequately minimized. The lower value of soft constraints i.e. 809 as compared to value of 2000 for hard constraints is attributed through treatment of measure of violation of soft constraints using Fuzzy Sets.

The final step encodes and evaluates manual solution for similar timetable problem that was already available. The manual solution was evaluated through same Fitness Function that was also used for FGH optimizer. The comparative results of manual solution , GA solution [29] and FGH solution are given in Table 7, where *objective value* is part of fitness value attributed to violation of soft constraints i.e., objectives. The *penalty value* is part of fitness value attributed to hard constraints. The *classroom hour gaps* is total number of hours within classroom schedules during which classrooms are unoccupied

and *teacher hour gaps* is total number of hours within each teacher's schedule during which teacher does not have class assignment.

As is obvious from Table 7, FGH optimizer manages to satisfy all hard constraints. It is also evident that solution produced by FGH satisfies soft constraints better than manual solution. FGH solution scores an objective value of 809 compared to 2286 of manual solution and 1107 of GA solution [26]. The value of 809 corresponds to only 5 *classroom hour gaps* and only 1 *teacher hour gap* compared to 90 and 98 of manual solution and 7 and 2 of GA solution [26] respectively. It seems like manual solution was outcome of focused effort to satisfy hard constraints, while soft constraints didn't received much importance. On other hand, FGH which has been treated with fuzzy membership functions and GA [26] algorithms are well developed concerning both hard and soft constraints.

The proposed FGH algorithm is implemented in Microsoft Visual C++ 6 under Windows XP on Intel machine having 2 GHz processor and 512 RAM. The algorithm is tested on data instances prepared by Socha data instances which are generated by generator written by Ben Paechter available on website: http://iridia.ulb.ac.be/~msampels/tt.data/. The data instances measure the performance of approaches related to UCTP and are prepared carefully to mimic real word UCTP at St. Xavier's College, Kolkata with different size and supersets of constraints. The data instances are classified into three classes as small, medium and large with respect to parameter value for each class. Experiments are performed on five small data instances, five medium data instances and one large data instance; they were tested using 7000 iterations as mentioned to section 4.2. Time consumed for each instance is approximately 20 minutes for each small instance and less than 2 hours for each medium and large instance. As FGH is basically a GA based algorithm with fuzzy fitness function it is worthwhile to compare it with other GA based heuristic algorithms for UCTP to illustrate its effectiveness. To demonstrate the significance of FGH algorithm, a comparative performance of execution times is made with respect to seven different GA based heuristics as given in Table 8. FGH algorithm takes an appreciable amount of time to generate satisfactory solution in comparison to other GA based heuristic solutions.

Table 5: Simulation Results for Standard Operators and Parameters

Setup	Mean Quality	Best Quality	Worst Quality
Standard	74192	61087	97073
1 (a)	72286	52046	90669
1 (b)	**67886**	**60009**	**75969**
2 (a)	72389	56966	82976
2 (b)	81690	65010	93024
3 (a)	61326	50024	75996
3 (b)	**61484**	**44026**	**77987**

4 (a)	**66282**	**42010**	**82997**
4 (b)	65090	52024	75999
5 (a)	70095	55998	82998
5 (b)	65887	52990	82884
6 (a)	65888	54027	87998
6 (b)	65186	55997	75987
7 (a)	71665	60995	80024
7 (b)	76882	52030	88987
8 (a)	**53490**	**45036**	**64995**
8 (b)	59680	44985	76996
9	68479	46046	91980
10	58278	47982	70999
11 (a)	50265	38978	66980
11 (b)	**52269**	**36975**	**62936**
11 (c)	57696	47660	69526
12 (a)	53277	38984	88024
12 (b)	**42036**	**26002**	**58982**
13	**29782**	**22982**	**35030**

Table 6: Simulation Results for Domain Specific Operators

Setup	**Mean Quality**	**Best Quality**	**Worst Quality**
Day Change	20886	14446	28686
Fix Teacher	18169	12421	28684
Fix Room	12996	9466	22989
Fix Day	6936	2809	11130

Table 7: Comparison between Manual and Fuzzy Genetic Heuristic Solution

Feature	**Manual Solution**	**Genetic Algorithm Solution** [26]	**Fuzzy Genetic Heuristic Solution**
Fitness	2286	2599	2809
Objective Value	2286	1107	809
Penalty Value	0	2000	2000
Number of Hard Constraints Violated	0	0	0
Number of Soft Constraints Violated	0	0	0
Classroom Hour Gaps	90	7	5
Teacher Hour Gaps	98	2	1

Table 8: Comparison of Execution Times (in minutes) of Fuzzy Genetic Heuristic Solution with other GA based Heuristic techniques

Datasets	GA1	GA2	GA3	GA4	GA5	GA6	GA7	FGH
Small1	11.55	15.79	16.60	17.45	14.56	16.62	18.32	19.76
Small2	11.59	15.86	16.64	17.46	14.57	16.64	18.33	19.90
Small3	11.62	15.96	16.66	17.47	14.59	16.66	18.34	19.86
Small4	11.64	15.98	16.69	17.50	14.60	16.67	18.35	19.89
Small5	11.69	15.99	16.86	17.52	14.69	16.69	18.37	19.89
Medium1	109.96	106.90	116.99	115.90	111.30	112.30	112.28	119.07
Medium2	104.86	109.50	107.84	107.86	86.32	80.32	79.86	119.56
Medium3	110.99	-	118.30	117.99	114.37	112.37	112.16	118.66
Medium4	-	100.79	104.56	115.57	114.54	112.50	112.37	117.96
Medium5	-	-	105.99	115.96	85.69	75.69	69.86	118.98
Large	-	-	-	-	116.32	119.30	119.55	119.99

In Table 8, the experiments are performed on different datasets viz., small, medium and large. In this process, the following different GA based heuristic abbreviations are used: GA1: Genetic Algorithm1 [16]; Genetic Algorithm2 [23]; GA3: Genetic Algorithm3 [52]; GA4: Genetic Algorithm4 [47]; GA5: Genetic Algorithm5 [58]; GA6: Genetic Algorithm6 [26]; GA7: Genetic Algorithm7 [34]; FGH: Fuzzy Genetic Heuristic

6 Conclusion

In this work FGH algorithm is presented for UCTP. The technique uses an indirect representation featuring event allocation priorities and invokes timetable builder routine for constructing the complete timetable. The algorithm incorporates number of techniques and domain specific heuristic local search operators to enhance search efficiency. The non-rigid soft constraints involved in problem are basically optimization objectives for search algorithm. Micro GA is also incorporated in the algorithm to improve performance of GA by improving the Fitness evaluation, such that the entire search space is reduced by Crossover alone and there is rapid convergence to sub-optimal solution. There is an inherent degree of uncertainty involved in objectives which comprises of different aspects of real life data. This uncertainty is tackled by formulating measure of violation parameter of soft constraint in fitness function using fuzzy membership functions. FGH algorithm has been applied on real world UCTP for which manual solutions are already available. It has been shown through extensive simulation that incorporating certain combinatorial and domain specific operators search efficiency of evolutionary algorithm is significantly enhanced. It may be mentioned that a big population is taken for GA implementation in this work; however, the micro setting of the population should be at least 40 to be 50 percent of population considered. By comparing FGH algorithm with manual solution it is evident that the technique satisfies all hard constraints of problem and achieves significantly better score in satisfying soft

constraints and thus its performance is superior. However, algorithm is computationally complex when compared to other different GA based benchmark heuristics. Further, to verify efficiency and robustness of algorithm, it should be tested on different real world timetable problems. The algorithm can also be adapted to solve other UCTP as well as to scheduling problems.

7 Future Work

In the process of developing UCTP through FGH algorithm, all hard and soft constraints are satisfied and significant results are obtained. However, computational time required is appreciably large. The future work entails in the development of well known heuristics viz., Neuro Fuzzy Genetic or Rough Fuzzy Genetic techniques which can reduce underlying computational complexity such that quality of solutions is greatly enhanced.

References

[1] S. Abdullah, E. K. Burke and B. McCollum, "A Hybrid Evolutionary Approach to the University Course Timetabling Problem", *Proceedings of the IEEE Congress on Evolutionary Computation*, Singapore, (2007).

[2] P. Adamidis and P. Arapakis, "Evolutionary Algorithms in Lecture Timetabling", *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, (1999), pp.1145-1151.

[3] H. Asmuni, E. K. Burke and J. Garibaldi, "Fuzzy Multiple Heuristic Ordering for Course Timetabling", *Proceedings of the 5th United Kingdom Workshop on Computational Intelligence*, London, (2005).

[4] M. Battarra, B. Golden and D. Vigo, "Tuning a Parametric Clarke-Wright Heuristic via a Genetic Algorithm", Università di Bologna, Dipartimento di Elettronica Informatica e Sistemistica, TR-DEIS OR.INGCE 2006/1R, (2006).

[5] S. C. Brailsford, C. N. Potts and B. M. Smith, "Constraint Satisfaction Problems: Algorithms and Applications", *European Journal of Operational Research*, Vol.119, (1999), pp.557 - 581.

[6] E. K. Burke and J. P. Newall, "A New Adaptive Heuristic Framework for Examination Timetabling Problems", University of Nottingham, *Working Group on Automated Timetabling*, TR–2002-1, (2002).

[7] E. K. Bruke and S. Petrovic, "Recent Research directions in Automated Timetabling", *European Journal of Operational Research*, Vol.140, No.2, (2002), pp.266-280.

[8] E. K. Bruke and J. P. Newall, "Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings", *Annals of Operations Research*, Vol.129, (2004), pp.107-134.

[9] E. K. Burke, B. McCollum and A. Meisels, "A Graph based Hyper Heuristic for Educational Timetabling Problems", *European Journal of Operational Research*, Vol.176, No.1, (2007), pp.177-192.

[10] A. Calaor, E. Hermosilla, Y. Augusta, and B. O. Corpus, "Parallel Hybrid Adventures with Simulated Annealing and Genetic Algorithms", *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks*, (2002).

[11] D. Campbell, "Harmonious and Achromatic Numbers and related Coloring Problems in Graph Theory," *M. Sc. Thesis*, University of Dundee, (2004).

[12] D. A. Coley, "*An introduction to Genetic Algorithms for Scientists and Engineers*," World Scientific Publishing Company, Singapore, (2001), pp.23-25.

[13] T. B. Cooper and J. H. Kingston, *The Complexity of Timetable Construction Problems*, Lecture Notes in Computer Science, Springer Verlag, Vol.1153, (1996), pp.283–295.

[14] D. Corne, H. S. Fang and C. Mellish, "Solving the Modular Exam Scheduling Problem with Genetic Algorithms", *Proceedings of the 6th International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Edinburgh, (1993).

[15] P. Cote, T. Wong and R. Sabouri, "Application of a Hybrid Multi-Objective Evolutionary Algorithm to the Uncapacitated Exam Proximity Problem", *Selected Papers from the 5th International Conference on the Practice and Theory of Automated Timetabling*, E. K. Bruke, M. Trick (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.3616, (2005), pp.151-168.

[16] M. Čupić, M. Golub and D. Jakobović, "Exam Timetabling using Genetic Algorithm", *Proceedings of ITI 31st International Conference on Information Technology Interfaces*, Croatia, (2009).

[17] D. De Werra, "An introduction to Timetabling", *European Journal of Operations Research*, Vol.19, (1985), pp.151-162.

[18] S. Deris, S. Omatu, and H. Ohta, "Timetable Planning using the Constraint-based Reasoning," *Computer and Operations Research*, Vol.27, (2000), pp.819-840.

[19] D. Dubois and H. Prade, *Possibility Theory*: *an Approach to Computerized Processing of Uncertainty*, New York, (1988).

[20] T. Duong and K. Lam, "Combining Constraint Programming and Simulated Annealing on University Exam Timetabling", *Proceedings of RIVF Conference*, Hanoi, Vietnam, (2004).

[21] Z. W. Geem, "School Bus Routing using Harmony Search", *Proceedings of Genetic and Evolutionary Computation Conference*, Washington, D.C., (2005).

[22] M. Gendreau and J. Potvin, "Tabu Search", *Introductory Tutorials in Optimization, Decision Support and Search Methodology*, E. K. Burke and G. Kendall (Editors), Springer Verlag, Chapter 6, (2005), pp.165-186.

[23] S. Ghaemi and M. T. Vakili, "Using a Genetic Algorithm Optimizer Tool to solve University Timetable Scheduling Problem", Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran, TR-2006-2, (2006).

[24] D. E. Goldberg, *Genetic Algorithms – in Search, Optimization and Machine Learning.* Pearson Education, Ninth Indian Reprint, New Delhi, (2008).

[25] J. F. Gonçalves and J. R. De Almeida, "A Hybrid Genetic Algorithm for Assembly Line Balancing", *Journal of Heuristics*, Vol.8, (2002), pp.629-642.

[26] P. Gupta, M. Bansal and H. Prakash, "Implementation of Timetable Problem using Genetic Algorithm", Department of Computer Science Engineering, Indian Institute of Technology, Kanpur, Project Report, (2006).

[27] S. A. Kazarlis, A. G. Bakirtzis and V. Petridis, "A Genetic Algorithm Solution to the Unit Commitment Problem", *IEEE Transactions on Power Systems*, Vol.11, No.1, (1996), pp.83-92.

[28] S. A. Kazarlis, S. Papadakis, J. Theocharis and V. Petridis, "Micro Genetic Algorithms as Generalized Hill Climbing Operators for Genetic Algorithm Optimization", *IEEE Transactions on Evolutionary Computation*, Vol.5, No.3, (2001), pp.204-217.

[29] S. A. Kazarlis, "Micro Genetic Algorithms as Generalized Hill Climbing Operators for GA Optimization of Combinatorial Problems–Application to Power Systems Scheduling", *Proceedings of the 4th Conference on Technology and Automation*, Thessaloniki, Greece, (2002), pp.300-305.

[30] G. Kendall and N. M. Hussain, "A Tabu Search Hyper Heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology", Lecture Notes in Computer Science, Springer Verlag, Vol.3616, (2005), pp.270-293.

[31] S. Khuri, T. Walters and Y. Sugono, "A Grouping Genetic Algorithm for Coloring the Edges of Graphs", *Proceedings of the ACM/SIGAPP Symposium on Applied Computing, ACM Press*, (2000), pp.422-427.

[32] G. Klir and T. Folger, *Fuzzy Sets*, *Uncertainty and Information*, Prentice Hall, New Jersey, (1988).

[33] A. Konar, *Computational Intelligence Principles, Techniques and Applications*, Springer Verlag, Netharlands, (2005).

[34] D. Kordalewski, C. Liu and K. Salvesen, "Solving an Exam Scheduling Problem Using a Genetic Algorithm", Department of Statistics, University of Toronto, Toronto, Canada, TR-2009-1, (2009).

[35] P. Kostuch, "The University Course Timetabling Problem with a 3-stage Approach", *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, (2004), pp.251-266.

[36] P. Kostuch, "The University Course Timetabling Problem with a 3-phase Approach", *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, E. K. Burke, M. Trick, (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.3616, (2005), pp.109–125.

[37] W. Legierski, "Constraint-based Techniques for the University Course Timetabling Problem", *CPDC*, (2005), pp.59-63.

[38] R. Lewis and B. Paechter, "New Crossover Operators for Timetabling with Evolutionary Algorithms", *5th International Conference on Recent Advances in Soft Computing*, Nottingham, England, (2004).

[39] R. Lewis and B. Paechter, "Application of the Grouping Genetic Algorithm to University Course Timetabling", *Evolutionary Computation in Combinatorial Optimization*, V. G. Raidl and J. Gottlieb, (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.3448, (2005), pp.144-153.

[40] R. Lewis, "Metaheuristics for University Course Timetabling" PhD Thesis, School of Computing, Napier University, Edinburgh, (2006).

[41] R. Lewis, B. Paechter and B. McCollum, "Post Enrolment based Course Timetabling: A description of the Problem Model used for Track Two of the Second International Timetabling", Cardiff University, Cardiff Business School, Accounting and Finance Section, (2007).

[42] R. Lewis, "A Survey of Metaheuristic based techniques for University Timetabling problems", *OR Spectrum*, Vol.30, (2008), pp.167-190.

[43] M. R. Malim, A. T. Khader and A Mustafa, "Artificial Immune Algorithms for University Timetabling", *Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling*, Czech Republic, (2006).

[44] C. Marco, B. Mauro and S. Krzysztof, "An Effective Hybrid Algorithm for University Course Timetabling", *Journal of Scheduling*, Vol.9, No.5, (2006), pp.403-432.

[45] R. Marti, H. Lourenco and M. Laguna, "Assigning Proctors to Exams with Scatter Search", Economics Working Paper Series No.534, Department of Economics and Business, Universitat Pompeu Fabr, (2001).

[46] L. M. Mooney, *Tabu Search Heuristics for Resource Scheduling with Course Scheduling Applications*, PhD Dissertation, Purdue University, (1991).

[47] J. M. Moreira, "A system for automatic construction of Exam Timetable using Genetic Algorithms", Revista de Estudos Politécnicos, *Polytechnical Studies Review*, Vol. 6, No. 9, Portugal, (2008).

[48] M. Omar, R. N. Ainon, and R. Zainuddin, "Using a Genetic Algorithm Optimizer Tool to generate good quality Timetables", *Proceedings of the 10th IEEE International Conference, Electronics, Circuits and Systems*, Vol.3, (2003), pp.1300-1303.

[49] B. Paechter, A. Cumming, M. G. Norman and H. Luchian, "Extensions to a Memetic Timetabling System", *Proceedings of the 1st International Conference on the Practice and Theory of Automated Timetabling*, (1995).

[50] V. Petridis, S. Kazarlis and A. Bakirtzis, "Varying Fitness Functions in Genetic Algorithm Constrained Optimization: The Cutting Stock and Unit Commitment Problems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol.28, No.5, (1998), pp.629-640.

[51] R. Qu and E. K. Burke, "Adaptive Decomposition and Construction for Examination Timetabling Problems", Multidisciplinary International Scheduling: Theory and Applications, P. Baptiste, G. Kendall, A. Munier-Kordon, F. Sourd, (Editors.), (2007), pp.418-425.

[52] R. Qu, E. K. Burke, B. McCollum, L. Merlot and S. Lee S, "A Survey of Search Methodologies and Automated Approaches for Examination Timetabling", Computer Science, TR-NOTTCS-2006-4, (2006).

[53] A. Ramani, F. Aloul, I. Markov and K. Sakallah, "Breaking instance independent symmetries in Exact Graph Coloring," *Journal of Artificial Intelligence Research*, Vol.26, (2006), pp.289-322.

[54] C. Reeves, "Genetic Algorithms", *Modern Heuristic Techniques for Combinatorial Problems*, V. J. Rayward-Smith (Editors), McGraw-Hill International, UK, (1995), pp.151-196.

[55] M. A. Saleh and P. Coddington, "A Comparison of Annealing techniques for Academic Course Scheduling", Lecture Notes in Computer Science, Springer Verlag, Vol. 1408, (1998), pp.92-114.

[56] A. Scholl and C. Becker, "State-of-the-art Exact and Heuristic solution procedures for Simple Assembly Line Balancing", *European Journal of Operation Research*, Vol.168, (2006), pp.666-693.

[57] B. Sigl, M. Golub, and V. Mornar, "Solving Timetable Scheduling Problem Using Genetic Algorithms", *25th International Conference Information Technology Interfaces*, Cavtat, Croatia, (2003).

[58] E. Singh, V. D. Joshi and N. Gupta, "Optimizing highly constrained Examination Timetable Problems", *Journal of Applied Mathematics, Statistics and Informatics*, Vol. 4, No. 2, (2008), pp.193 – 197.

[59] S. Slim and M. Marte, "University Timetabling using Constraint Handling Rules", *Journees Phrancophones de Programmation*, par contraintes, Nates, France, (1998).

[60] K. Socha, J. Knowles and M. Samples, "A Max-Min Ant System for the University Course Timetabling Problem", *Proceedings of the 3rd International Workshop on Ant Algorithms*, Lecture Notes in Computer Science, Springer Verlag, Vol.2463, (2002), pp.1-13.

[61] S. O. Tasan and S. Tunali, "A Review of the Current Applications of Genetic Algorithms in Assembly Line Balancing", *Journal of Intelligent Manufacturing*, Vol.19, (2008), pp.49-69.

[62] N. D. Thanh, "Solving Timetabling Problem using Genetic and Heuristic Algorithms", *Proceedings of 8th ACIS International Conference*, (2007).

[63] M. Tuga, R. Berretta and A. Mendes, "A Hybrid Simulated Annealing with Kempe Chain Neighborhood for the University Timetabling Problem", *Computer and Information Science*, (2007).

[64] G. White, B. Xie and S. Zonjic, "Using Tabu Search with Longer Term Memory and Relaxation to create Examination Timetables", *European Journal of Operational Research*, Vol.153, No.16, (2004), pp.80-91.

[65] Wren, Scheduling, "Timetabling and Rostering – A Special Relationship!", *The Practice and Theory of Automated Timetabling*: *Selected Papers from the 1st Int'l Conf. on the Practice and Theory of Automated Timetabling*, E. K. Burke, P. Ross (Editors), Lecture Notes in Computer Science, Springer Verlag, Vol.1153, (1996), pp.46-75.

[66] L. A. Zadeh, "Fuzzy Sets", *Information and Control*, Vol.8, (1965), pp.338–353.