IZMIR UNIVERSITY OF ECONOMICS

SENIOR PROJECT

CE 497

A GENETIC ALGORITHM FOR UNIVERSITY COURSE TIMETABLING PROBLEM

Prepared By

Barış Alp Yuncu 20110602039

Barış Başaran 20110602004

Deniz Dölek 20120602010

Instructor

- Doç. Dr .Cem Evrendilek

GROUP NAME

Cem and 3 Musketeers

13 January 2016

A GENETIC ALGORITHM FOR UNIVERSITY COURSE TIMETABLING PROBLEM

Table of Contents

ABSTRACT

University course timetabling problem (UCTP) is basicly a set of meetings in time. A meeting is combination of resources which might be classes, students, teachers etc. UCTP is an NP-hard combinatorial optimization problem which lacks analytical solution methods. The purpose is to schedule teachers and classes to fully utilize available resources. This problem arises in the process of trying to allocate elements according to a set of, so-called, hard and soft constraints using limited resources This problem is generally encountered in educational institutions, but it is also possible to encounter with this problem in many other industrial areas. We conducted a study on the performance of genetic algorithm in designing institutional lecture time table, using empirical data of Izmir University of Economics. The study was focused on assessing the effectiveness of the algorithm given a number of hard and soft constraints and a limited number of resources. The algorithm was implemented in Java. We aim to provide a graphical user interface and in order to do that we created a web-site which requires a user log-in so that an undefined user might not be able to reach universities data. The purpose of the web site is to provide quick access from anywhere and everywhere to the system.

INTRODUCTION

Scheduling is a process or a way of organizing time according to arrangement of work order plan. It also means a list or activity table or an activity plan with a detailed execution time [2]. Timetabling is one of the common scheduling problems, which can be described as the allocation of resources with factors under pre-defined constraints so that it maximizes the possibility of allocation or minimizes the violation of constraints [3].

Timetabling is an NP-hard problem.[4] It can be defined as scheduling of certain number of lectures, which are to be attended by a specific group of students and given by a teacher. Also it must be suitable for a denite period of time. The creation of a timetable itself is a very important and practical task.

The problem was first studied by Gotlieb [5], who formulated a class-teacher timetabling problem by considering that each lecture contains one group of student, one teacher and any number of times which might be chosen freely. Population-based algorithms, particularly genetic algorithms have been the most common solution in recent years. Genetic Algorithm (GA) is a powerful way to solve complex search problems[6]. GA provide us to find globally optimal solutions even in the most complex of search spaces.[7]

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 [4] of combinatorial nature and no optimal algorithm is known which generates solution within reasonable time. These problems are mainly classified as constraint satisfaction problems. There are number of versions of UCTP differing from one university to another. Due to complexity of the problem, most of work done concentrates on heuristic algorithms which try to find good approximate solutions [11], [12], [13], [14], [15], [16], [17], [18], [19]. Some of these include **Genetic Algorithms (GA)** [20], [21], [22], [23], **Tabu Search** [24], [25], **Simulated Annealing** [26], [27] and recently used **Scatter Search** [28] 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.

Brief History of Genetic Algorithms

The first examples of genetic algorithms appeared in the late 1950s and early 1960s, programmed on computers by evolutionary biologists. First implementations of genetic algorithms were not include functions such as crossover and mutation.

In 1962, John Holland's was the first to explicitly propose crossover and other recombination operators[8]. However, the seminal work in the field of genetic algorithms came in 1975, with the publication of the book Adaptation in Natural and Artificial Systems. This book was the first to systematically present the concept of adaptive digital systems using mutation, selection and crossover, simulating processes of biological evolution, as a problem-solving strategy[9]. At first, these applications were mainly theoretical. Today, evolutionary computation is a thriving field, and genetic algorithms are "solving problems of everyday interest"[10].

Project Scope

- Creating a user friendly web site and a web service which will help users to build their own time table, using genetic algorithm, according to information given by them. Applicability of the project will be tested using academic programs of Izmir University of Economics.
- Web site should be designed as simple as possible. User should be guided in order to prevent incorrect entries. User should give requested informations such as name of courses, lecturers etc. by using web site. If user couldn't manage to use web site correctly, disordered data will be sent to web service and it will decrease the quality of solution. Therefore web site part is important as web service and algorithm part.
- We are going to do a detailed research about solutions of lecture time tabling problem that implemets genetic algorithms from the past to the present.
- Our aim is creating the web site and web service which uses Genetic Algoritms to solve lecture timetabling problem by using genetic algorithms, we will also try to seek out an improvement in the quality of solution to lecture timetabling via genetic algorithms.

What software, hardware, or tools we use?

We have implemented genetic algorithm system in Java. Apache Tomcat used as web server. The excel file ,which will be taken as input from web site, will be the source of data. Data will be stored in mysql database. We choose Eclipse Java Mars as our IDE.

PROJECT DETAILS

Genetic Algorithm Part

GenerateTimetableServlet is the most important Servlet of our system. It generates timetable by using Genetic Algorithm. The purpose of our genetic algorithm is to solve university course timetabling problem with satisfying all hard and soft constarints.This servlet uses six different classes to execute the genetic algorithm.

- Population.java = This class corresponds to set of all possible solutions. For instance if population size is one hundread, it means this population includes one hundread different possible solution.
- Individual.java = This class corresponds to one timetable. Every timetable is a part of the solution set of population object. Timetable means one solution but this solution may not fit to all hard and soft constraints.
- Course.java = Every individual has one choromose which is a course object list. It means Course.java corresponds to gene of this chromosome.
- Lecturer.java = Every course has to include one lecturer. This class is assigned to Courses by Individual object.
- Room.java = Every course has to include one room. This class is assigned to Courses by Individual object.
- GeneticAlgorithm.java = This class includes functions which are the essential part of genetic algorithm such as selection,mutation,crossover etc...

Brief Summary of Genetic Algorithm

In our system, genetic algorithm creates set of timetables randomly and evaulates their fitness according to the hard and soft constraints. Afterwards it generates a new population by evaluating the old population. This process continues until the best solution found which satisfies all of the hard constraints and majority of soft constraints. Our system uses tournament selection function, uniform crossover, simple mutation function as basic operators. Furthermore our purpose is to test different those operators and to study on their effectiveness in order to build a fast and feasible solution.

Definitions

Some important definitions:

- Gene: a Gene is a part of solution. It contains some data.
- Chromosome: a Chromosome (or individual) is a solution to the given problem, composed by genes.
- Population: set of solutions.
- Soft constraints: constraints that may be breached
- Hard constraints: constraints that must be satisfied
- Generation: a “round” of the algorithm, with a new population, after evolution.

Steps of the GA

Choose **initial Population**
Repeat

Evaluate population

Select best-evaluated individuals for reproduction

Apply **crossover**/**mutation** reproduction

Set **new population**

Until ending condition (when a individual get the desired evaluation or a specified number of generations has been done).

HARD CONSTRAINTS

A hard constraint is a constraint which can not be violated. There should not be any violations in order to produce the correct solution.

- Each class is scheduled to take place in a specific period.
- Neither a class nor a teacher nor a room is assigned to more than one lesson in the same time-slot.
- No omission of classes in the timetable.
- All allocated rooms are large enough.
- Specific room requirements must be taken into considerations(i.e. lab).
- Pre-scheduled lessons must be scheduled to specific time-slot(i.e. liberal classes/rooms).
- All subjects are evenly distributed.
- Rooms should be just large enough to meet the requirements.

SOFT CONSTRAINTS

A soft constraint is a constraint which may be violated. It should be satisfied in order to obtain an optimal solution but the timetable is correct even if theses constraints are not satisfied.

- Neither students nor teachers like timetables with a lot of empty slots(empty periods between lessons).
- Each kind of event should be spread evenly.
- Teacher's preferences.
- Lecture rooms should be close to the host departments.
- Lectures of N'th year and N+1 year class's mandatory lectures should not be in the same time-slot.

Details of Algorithm

Initialisation

GenerateTimetableServlet receives name of TimeTable (ex. Bahar_2016), mutation rate, population size as parameter and executes genetic algorithm. If mutation rate and population size is given by user, this values will be set. else default values are 0.001 for mutation rate and 50 for population size. Afterwards GenerateTimetableServlet invokes the parameterised constructor of population object with parameter population size. When servlet invokes the parameterised constructor, the chain of events will be initialized.

Parameterised constructor of population object establishes connection with database named "Timetable Manager". It requests these data;

- Lecturer informations.
- Course informations.
- Room informations.

System connects to database and receives all informations at population creation stage because establishing mysql connection during creation of an individual(Timetable) object dramaticaly decreases the performance of the systems. For instance if population size is 100 and we establish connection at individual instead of population object, the system will execute extra 99 times of sql command which is nothing but lose of time. Afterwards, parameterised

constructor invokes parameterised constructor of individual object as much as population size which is indicated by user. The parameters are the data which is received from database "Timetable Manager".When population object invokes the parameterised constructor of individual object , another chain of events will be initialized.

Parameterised constructor of individual object starts a loop to create lecturer objects one by one. It invokes parameterise constructor of lecturer object by sending parameters, which are received from population object. This constructor does not execute any special events. It only sets parameters received to the corresponding values. For instance constructor will set parameter "name" to value "Lecturer name". When lecturer objects are created, individual objects constructor starts another loop to create room objects one by one. Room object is very similar to lecturer object. Due to this situation, similar steps will be executed for every room object which is going to be created according to the information received from population object. Room object and Lecturer object are similar to each other because they are both part of course object, they have same type of constructors. Their values are different from each other.

Individual object has chromosome which represents the unique solution to the problem. Choromosome structure is represented as course objects list in individual.java. As a reminder, individual objects is timetable which is going to be generated. Each gene of this chromosome represents a single course in timetable.

After lecturer and room objects are ready, the constructor of individual object starts final loop to create genes of chorosome, courses of timetable. It invokes the parameterised constructor of course object for every gene of chromosome. Parameters are :

- Some informations received from population object : Most of these informations are directly given to the parameterised constructor of course object because there is no possibility to change this type of information. For instance the name of course, department of course have to given directly to the course object as parameter because these can not be altered during the generation process.
- Lecturer object : In our program, every course has a lecturer who is already assigned. This information is received from database but instead of giving lecturer information, constructor of course object expects the lecturer object

itself. The system finds lecturer object accourding to the information which is received by population object and sends this object to constructor as parameter.

- Room object : In our system, some of the rooms may be pre-fixed.
    - If course object is going to have pre defined classroom or lab program finds the respected room object accourding to data which is received by population object and sets the object.
    - Else it selects one classroom randomly and sets this room to that course.
- Time slots : Constructor of individual object generates random time slots for every course object which is going to be created. Generates two integer values. One of the integers represents the day of course. For instance “1” refers to Monday and the other one represents start time of course. For instance “1”refers to 9.00 am.

The Structure to Detect Constraint Violations

The most important problem that causes violation of soft and hard constraints is double booking problem. In order to avoid from this problem lecturer, room and individual objects use hashmaps to detect violations. When a course object is created, function of specific object is called to fill the hashmap of this object.

For instance;

- a course named “ce 407” is created
- The system generates key from duration ,start time and day of the course to fill hashmap of lecturer and room objects.
- Individual objects hashmap key generation process requires one more parameter which is department of course.

When the key is generated, the system upgrades number at the specific part of hashmap, which is reached by the key. By the help of this approach, controlling double booking executes fast and effective. Controlling hashmaps occurs when the system needs to calculate violations. If the value at specific part of hashmap is more than 1, it means double booking violation is detected. Other types of violations does not needs any special

representation, simply comparing result with the value, which has to be. For instance, assume lecturer "Ahmet" is created and this lecturer has available days parameter, which is "Salı". The program is going to compare the lecturers available day with the day value of course which is assigned to this lecturer.

Course objects parameterised constructor sets parameters to the values corresponding to these parameters and invokes specific functions to fill hashmap of lecturer and room objects. Furthermore, after the course is created, the function to fill individual objects hashmap will be invoked and specific part of hashmap will be modified. When each course object had been created and hashmaps modified, the individual would be ready. Then population is ready if all individuals are created.

Finding the Optimum Solution and fitness calculation

Creation of the first population is the first step of generating a feasible solution by using genetic algorithm. Afterwards system calls specific function to find best solution of population. Population objects has balanced binary search tree to find best individual(Timetable) of population. Every individual has a function to calculate the cost of itself. The cost value of individual increases as much as the violations it contains. For instance the cost value "0" means this solution is a desired solution.

Population object requests cost from every individual object. Every individual object starts to control their hashmaps by generating special key for all course object and control the hashmap area which corresponds to the generated key. Hashmaps are used for detecting double booking violations. System detects other type of violations by comparing the generated value with the value which has to be. For instance, assume lecturer "Ahmet" is created and this lecturer has available days property, which is "Salı". The program is going to compare the lecturers available day with the day value of course which is assigned to this lecturer. After every individual calculates its own cost, the function of population stores them into the balanced binary search tree and finds the best individual of population by the help of binary tree and returns the cost of this individual.

Evolving Population to Create Acceptable Solution

After the cost of the best individual received from the population, the system checks this value. If this value is equal to “0” it means the solution already exists. Else GenerateTimetableServlet starts a loop which continues until desired solution found. Inside of loop, EvaluatePopulation function is going to be executed. This function is the main function of GeneticAlgorithm.java. GeneticAlgorithm.java includes genetic algorithm specific functions which are :

- Evolve Population Function **:** This function takes a population and by using this population it creates new population. It creates an empty population inside and stores the least costed tables object to the position '0'. Then it starts the loop. At every loop it calls tournament selection function twice. After that it invokes crossover function. It takes the result of crossover and invokes mutate function. Lastly it invokes fillthetables function and save the tables object to the specific location. After loop is over it invokes calculate AllCostOfTables function of population and returns the new created population.
- Tournament Selection Function**:** It takes list of individuals (Timetables) from the population and the size of this population. This function selects three of them randomly and returns the least costed tables object. Tournament selection is the default selection method of our genetic algorithm system.
- Uniform CrossOver Function**:** This function takes two individual (Timetable) object(x and y)and creates a new individual object and fills this object by using these parameters. It creates loop, which goes through chromosome(Course object array, Course object = gene) and inside this loop it invokes Math.random(). If Math.random returns value that is smaller than 0.5, the new individual object takes parent x's gene and puts this gene to the same slot in chromosome, else the new individual object parent y's gene and puts this gene. Afterwards, same steps will be executed for the next gene until all genes are traversed. Moreover, a gene in chromosome is in always in same order for every individual. For instance assume the first gene is course object with name “cloud computing” it will be the same for every individual’s chromosome. Finally new individual is became the combination of parent individuals x and y. This new individual is going to be returned as a response.

- Mutate Function : This function takes an individual object as parameter. Then function invokes Math.random. If math random is smaller than mutation rate, which is received from user or set default as 0.001, this function changes the day, start hour and room value of that tables object, else it does nothing. Room value is going to be changed if it is not pre-defined. Finally, it returns the individual object as a response.

Evaluate Population function creates an empty population object by calling the default constructor. It stores the best individual, which has least cost, to the position "0" of population. This method is called as elitism and it is useful to save best individual to the next generation. This method is used because if we do not use elitism, the evolving population process may create worse population than previous population and this situation decreases the performance of genetic algorithm. Afterwards, evaluate population starts a loop to fill the rest of new population, by creating new individuals to this new population. To create new individual, this function invokes selection function twice, receives two parents and crossovers them. After crossover returns the new child individual, it invokes mutation function.

The purpose of mutation function is increase the variety of population. Furthermore it calculates the cost of this individual and stores the new individual to the new population. These steps are executed until new population is ready. Then evaluate population function returns this new population as response.

This evolve process is going to be executed until the system finds the acceptable result. When it does, it creates a mysql connection to the database Timetable Manager and stores individual, which is an acceptable timetable result, to database and sends a sign which indicates generating timetable process is succesful. Else it sends another sign which claims failure.

TESTS

There are lots of detailed research about solving university course timetabling problem with genetic algorithm and these researches includes unique solutions to solve this problem[29][30][31][32]. Details of algorithm part uses popular genetic algorithm operators (uniform crossover, tournament selection) and it is the default algorithm of our system. Furthermore, the program is going to make changes on rate values, genetic algorithm operators, cost calculation system and observes the effects of this changes on performance.The main purpose of making tests on system is create system which

demonstrates best performance on solving university course timetabling problem. Tests of system is not completed. A group of tests can be added after this report is delivered or next semester. Tests instances are :

- Applying another popular crossover techniques: This test applies one-point crossover, multi-point crossover and half uniform crossover methods instead of uniform crossover method, which is default crossover type of our system.
- Applying roulette wheel selection instead of tournament selection.
- Changing the rates of crossover and mutation and observing the performance.
- Applying asynchronous island model on our system : It is an example of multi-threaded approach on genetic algorithm[29].
- Some changes may effect systems performance when they are applied together. Therefore the every combination of tests will be applied as well.

Web Part

Our system has three kind of user: Root, Admin and regular user. Root user has following authorisation :

- Create & Manage Data which is used for generating timetable.
- Generate & Modify Timetable.
- Manage administration authorisation of users.
- Manage own account informations.

Admin user has following authorisation :

- Create & Manage Data which is used for generating timetable.
- Generate & Modify Timetable.
- Manage own account informations.

Regular user has following authorisation :

- View Data which is generated by admin or root.
- View Timetable which is generated by admin or root.
- Manage own account informations.

Index.html welcomes user. In index.html, there are a title and a basic explanation about timetable manager. Login and Sign up buttons are located on this html page. If the user has no account before, new account must be created by using Sign up part, else the user should use

login part. When user clicks the Sign up button, username, password, name, surname, email and department fields will appear. Then user fills these areas and signupServlet completes sign up process. After sign up process is succesful, the user is directed to mainpage.html. At sign up part only regular user account can be created. Login part is similar to sign up part. When user clicks login button, username and password fields will appear. Then user fills these areas and loginServlet starts to execute. LoginServlet will return a value accourding to the authorisation of user and the user is directed to the one of these pages accourding to the authorisation:

- Root user will be directed to rootpage.html.
- Admin user will be directed to adminpage.html.
- Regular user will be directed to mainpage.html.

In mainpage.html, there are three buttons: view table, view data, account management.

- The user is capable of displaying the generated timetable and searching through the timetable with the view table button.
- The user is capable of displaying the generated data for timetable and searching through them with the view data button.
- The user is capable of updating account informations by clicking Account Management button. When clicking this button, username, old password, new password, name, surname, email and department fields are appeared. Then user fills these fields and updateaccountServlet handles the update process. If update process is succesful, The user will be directed to the index.html to login again. Else system requests user to enter these areas again. In addition, if user enters wrong value to old password field, update process will not be handled. If user does not enters any value to new password field, the password of user is not going to be changed.

In rootpage.html, the root come across with four buttons: Data Management, Timetable Management, Account Management, Authorisation Management.

Data Management provides to create the data and manage the data. When the root clicks the create data button, two possible fields are formed: Create Courses, Create Lecturers. The root can upload the excel files which are conformed with pre-designed format. When user gives specific excel files, uploadcoursesServler or uploadlecturersServlet handles the upload

request. Manage data button lists the lecturers and course identifiers. Afterwards the root selects lecturer or course to manage process. BringdataServlet manages request and bring all lecturer and course identifiers. Finally, the root may be able to update changes on selected object or delete the selected object. UpdatecoursesServlet, updatelecturersServlet handles update processes and DeletecoursesServlet, DeletelecturerServlet handles delete processes.

Timetable Management provides to generate the timetable and manage it. When the root clicks the Timetable Management button, two fields are formed: Generate the timetable , Manage the timetable. When user gives a name to timetable, mutation rates, population size and clicks button generate, GeneratetimetableServlet handles this request, generates timetable and informs user about process. When user selects to manage timetable which is generated before and select specifications ,which will generate a view of timetable, UpdateModificationstoTimetableServlet handles this request.

The root is capable of updating account informations by clicking Account Management button. It is similar with regular user's Account Management. When clicking this button, username, old password, new password, name, surname, email and department fields are appear.

Authorisation Management provides to update all users' authorisations on system. When the root clicks this button, BringallusersServlet handles this request and the root is able to reach authorisation informations of all other users and change them.If user wants to change authorisation of any user, ChangeAuthorityServlet handles the request.

In adminpage.html, the admin come across with three buttons: Data Management, Timetable Management and Account Management. These fields are exactly same as rootpage.html.

Servlets

SignupServlet : This servlet provides sign up process that receives username & password & name & surname & email & department as parameter, creates mysql connection to database which is named timetablemanager. This servlet checks every parameter to avoid possible SQL injection attack. Afterwards signupServlet uses mysql connection to generate new user. Two possible failure may occur during sign up process. These are :

- Creating a username which already exists.
- Empty password field.

If there is no failure during sign up process, servlet sends a sign which indicates sign up process is succesful. Else it sends another sign which claims failure.

LoginServlet : This servlet is used for login process. Receives username & password as parameter, creates mysql connection to database, which is named timetablemanager. Furthermore this servlet retrieves password by using username parameter. It compares password retrieved from database with password retrieved as parameter. If these two are equal, loginServlet sends a sign which indicates login process is succesful else it sends another sign which claims failure.

BringAccountInfoServlet : This servlet is used for bringing the user informations to the webpage. Receives username as parameter and creates mysql connection to database named “timetablemanager”. This servlet checks username parameter to avoid possible SQL injection attack. Moreover bringaccountinfoServlet receives account informations by using parameter of username and sends html tags with informations of user.

UpdateAccountServlet : This servlet will be invoked when a user wants to update his/her account. Receives old username & new username & old password & new password & name & surname & email & department as parameter, creates mysql connection to database, which is named “timetablemanager”. This servlet checks every parameter to avoid possible SQL injection attack. UpdateAccountServlet uses mysql connection to update user information. In addition, it has control over new password area. If new password area is empty, the old password is valid as password. If the user change the password, it will be updated and the new password is valid.

UploadlecturersServlet, UploadcoursesServlet : These servlets are invoked when the users send request to update lecturers or courses in database by using an excel file. These servlets receive file object as parameter, creates mysql connection to database. They start to update database by using a specific excel file and mysql connection. The excel file should be

in pre-designed format otherwise this servlets will not work properly. These servlets may raise error when the file format not matched with the previously determined format. Finally, uploadlecturersServlet and uploadcoursesServlet sends html tags with information of process and link to previous page.

BringdataServlet : This servlet is invoked when the users send request to manage course and lecturer data which reside in database. This servlet does not receive any parameter. Furthermore, it sends response as html tags with information of all lecturers' name and all course identifiers.

BringcourseServlet, BringlecturerServlet : These servlets are requested when the users want to bring all information of the course and the lecturer. BringcourseServlet receives course id as parameter, BringlecturerServlet receives lecturerid as parameter and they create mysql connection to database. Afterwards they send response as html tags with information of selected course or selected lecturer.

DeletecourseServlet, DeletelecturerServlet : The purpose of these Servlets is to delete course and lecturer objects when requested. They receive course id and lecturer id as parameter and create mysql connection to database. Moreover, they send request to database for deletion of selected object. If there is no failure during deletion process, servlets send a sign which indicates deletion process is succesful. Else they send another sign which claims failure.

UpdatecourseServlet : This servlet receives course id & course code & course name & course section & maximum number of students & semester of course & department & total weekly hour of course & lecturer and course type as parameter. It creates a mysql connection to database and . checks every parameter to avoid possible SQL injection attack .Afterwards, updatecourseServlet updates changes to the database. If there is no failure during update process, servlets send a sign which indicates update process is succesful. Else they send another sign which means failure.

UpdatelecturerServlet : This servlet receives lecturer id & lecturer name & available days of lecturer and fixed hours of lecturer as parameter. It creates a mysql connection to database and . checks every parameter to avoid possible SQL injection attack .Afterwards, updatelecturerServlet updates changes to the database. If there is no failure during update process, servlets send a sign which indicates update process is succesful. Else they send another sign which means failure.

BringallusersServlet : This servlet requested when user wants to manage authorisation of all other users. This servlet does not receive any parameter. It receives all users authorisation informations.Furthermore, bringallusersServlet sends html tags with authorisation information of users as response.

ChangeAuthorityServlet : This servlet requested when root wants to change authorisation of selected users. This servlet receives username & new authority as parameter and creates mysql connection to database. Afterwards it updates authority value of selected user. If there is no failure during update process, servlet sends a sign which indicates update process is succesful. Else it sends another sign which means failure.

GenearateTimetableServlet : This is the most important servlet of this system. It receives name of timetable as parameter. Then it starts to execute genetic algorithm to generate suitable timetable. Afterwards, generateTimetableServlet saves result to database. If there is no failure during generation process, servlet inform user by giving a sign which indicates timetable generation process is succesful. Else it gives another sign which means timetable generation process is failed.

BringTimetableServlet : This servlet is invoked when the user wants to manage and see specific part of generated Timetable. BringTimetableServlet receives parameters name of timetable,which presents id of timetable, and specifications which is used for creating specific

view of timetable. Furthermore the servlet retrieves data of timetable according to specifications that are received, and sends html tags with information of this data as response.

UpdateModificationstoTimetableServlet : This servlet updates the changes on generated timetable. This servlet retrieves name of timetable and every changes as parameter, creates mysql connection to database which is named timetablemanager. This servlet checks every parameter to avoid possible SQL injection attack. Afterwards it updates parameters, which are retrieved,on previously generated timetable. If there is no failure during update process, servlet inform user by giving a sign which indicates timetable update process is succesful. Else it gives another sign which means this process is failed.

Timeline

1st semester

- 1. Project selection and team formation
- 2. Beginning of the project
- 4. Project proposal submission
- 8. Background research and requirements specification document submission
- 14. Completion of System Design
- 15. Oral presentation

2nd semester

- 2. Completion of web part of the project
- 4 completion of GA implementation
- 5 beginning of the tests
- 10. Testing (validation and verification) document submission
- 14 Implementation document submission
- 15 Delivery and oral presentation

FUNCTIONAL AND NON-FUNCTIONAL REQUIREMENTS

Non-Functional Requirement ID	Requirement Statment	Must/ Want	Comment
WebSiteStructure001	The web site shall have a homepage.	Must	
WebSiteStructure002	Homepage shall have a login and sign up and after login	Must	After login homepage shall direct user to mainpage
WebSiteStructure003	Homepage shall give a brief information about the way that system works.	Must	
WebSiteStructure004	Web site shall have a mainpage	Must	
Authorization001	There should be a user tree. A root user and children users.	Must	Root user can give priviledge to child users for acces to stored data.
Authorization002	Only root user shoul be allowed to generate timetables and insert excel data to DB	Must	
Authorization003	TimeTable generation process starts with root users command	Want	
Authorization004	Only authorized users shall be able to give excel file for insertion of the required data.	Must	Excel file must be in right format.
DataStorage001	Stored data in database should be modifiable.	Must	User shall be able to manipulate the datastored in database.
DataStorage002	Both output of generated timetable and the data stored in database should be queried.	Must	So that user can display data partially.
UserData001	User shall be able to change his/her password and e-mail adress.	Want	
UserGuide01	Help text will be provided both in English and Turkish.	Want	

Security001	User password should be encrypted before storing in	Must	Password encrypted with
GA002	Random variables needed for mutation and crossover	Must	
GA003	Some Violation of constraints shall be checked with hashmaps	Must	
Security003	System should destroy sessions if user inactive for a certain amount of time	Must	
Security004	Atomic transaction model should be used for accesing data stored in database.	Must	
DataStorage001	Data which is read from excel should be stored in database	Must	In our case derby database used for this purpose
DataStorage002	System should update stored data	Must	
DataStorage003	Stored data should be ready for creation of timetable at any time.	Must	
SystemStructure 001	System uses genetic algorithm to generate a timetable	Must	
SystemStructure 002	System creates several object to support GA Population,individual,course,lecturer, department, room	Must	Each individual has specific solution and this solution is consist of lecturers, courses, departments, rooms etc...
SystemStructure 003	System implements specific hashmaps for lecturer, department and room objects to increase speed of evaluation step.	Must	
SystemStructure 004	System should support multi user access concurrently	Must	
Error001	If any error occurs system should log detailed information about that error.	Must	System should both send information about error to user and administrator.
GA001	Genetic Algorithm has 5 main steps; initialization, evaluation,selection,crossover,mutation	Must	

GA004	Population class should have specific functions to retrieve data from database.	Must	
Validation001	System should find violations caused by modifications and prevent them	Must	
Validation002	Insertion, update, delete operations which might be done by user should be validated by system.	Must	
Performance001	System should find generated timetable within a resonable time	Must	
Performance002	System should create ajax request for each user request which is generated in mainpage.	Must	Import data, view data, change password and e-mail requests etc.

CONCLUSION

This paper has presented genetic algorithm for solving timetable scheduling problem and a web site that includes a timetable manager. We conducted a study on the performance of genetic algorithm in designing timetable. The study is focused on assessing the effectiveness of the genetic algorithm given a number of hard constraints and soft constraints with a limited number of resources.

In first semester, we did the background research. We had completed the system design and implemented GA's core functions. Also created a database and a web site which is currently in development stage, We create a dynamic web site by using Java, html, CSS, Javascript and Ajax. We use servlets and Apache Tomcat web server.

In second semester, we implemented genetic algorithm, test and analyze its core functions and complete the design of the web site. We reached our goal which is to build a fast and feasible timetable manager and to create a website that can be used by anyone.

APPENDIX

USE CASE DESCRIPTION

Use-case #1

Title: A Genetic Algorithm For University Course Timetabling Problem

Use Case Name: View the Generated Timetable

Description: User view the timetable which is generated by the admin previously. Timetable is developed with Genetic Algorithm. Timetabling problem is basicly a set of meetings in time. A meeting is combination of resources which might be classes, students, teachers etc.

Actors: Regular User, Root or Admin

Preconditions:

1) User press the Sign Up button and fill the necessary fields.

2) User password should be encrypted before storing in DB

3) User login to the web site.

Postconditions:

1) Display all timetable

Normal Flow:

1. Open the web site and login to system.
2. Press View Data button
3. Check the results and display lecturers and courses.

Alternative Flow:

3a. Searching the courses and the lecturers through the timetable

Use-case #2

Title: A Genetic Algorithm For University Course Timetabling Problem

Use Case Name: Generate a Timetable

Description: Generate a timetable with using the timetable manager. The timetable manager is developed with Genetic Algorithm. Timetabling problem is basicly a set of meetings in time. A meeting is combination of resources which might be classes, students, teachers etc.

Actor: Root or Admin

Preconditions:

1) Root enters the spesific username and password
2) Root's password should be encrypted before storing in DB
3) Root login to the web site.

Postconditions:

1) Timetable is generated succesfully.

Normal Flow:

1. Open the web site and login to system.
2. The root clicks the Data Management button upload the excel files which are conformed with pre-designed format for both lecturers and courses.
3. Click the generate button and start the generation of timetable.
4. Check the results
5. Log out of the web site

Alternative Flow:

4a. If does not satisfied from results go back to step 3

Use-case #3

Title: A Genetic Algorithm For University Course Timetabling Problem

Use Case Name: Update account information

Description: User can update the account informations. Username, password, name, surname, email and department are modifiable features.

Actor: User, Root or Admin

Preconditions:

1) User enters the spesific username and password

2) User's password should be encrypted before storing in DB

3) User login to the web site.

Postconditions:

1) The account is updated succesfully

Normal Flow:

1. Open the web site and login to system.
2. Click the Account Management button.
3. Fill the username, old password, new password, name, surname, email and department fields.
4. The user will be directed to the home page to login again.

Alternative Flow:

4a. If the problem occurs, system requests user to enter these areas again.

Exception:

3a. If user enters wrong value to old password field, update process will not be handled.

3b. If user does not enters any value to new password field, the password of user is not going to be changed.

References

1. A. Jain, S. Jain, P. K. Chandle, "Formulation of Genetic Algorithm to Generate Good Quality Course Timetable" (August 2010).
2. D. Sugono, "Kamus Besar Bahasa Indonesia", Pusat Pembinaan dan Pengembangan Bahasa Indonesia, Departemen Pendidikan dan Kebudayaan, Balai Pustaka PN, 1993.
3. N. D Thanh Solving timetabling problem using genetic and heuristics algorithms Journal of Scheduling, 9(5): 403–432, 2006.

4. A. Jain, S. Jain, P. K. Chandle, “Formulation of Genetic Algorithm to Generate Good Quality Course Timetable” (August 2010).
5. Gotlieb, C.C., The construction of class-teacher timetables. In Proceedings of IFIP Congress, North-Holland Pub. Co., Amsterdam, 1962, 73-77.
6. S. Lukas, P. Yugopuspito and H. Asali, “Solving assignment problem by genetic algorithms using Cycle Crossover”, Universitas Pelita Harapan Computer Science Journal.
7. Joshua Poh-Onn Fan, Graham K. Winley. (2008). A Heuristic Search Algorithm for Flow-Shop Scheduling”, August 2011
8. Development of a University Timetable Automation System 08CG07800 by Oyebanjo Samuel Adejuwon (May 2012).
9. Mitchell Melanie (1996). An Introduction to Genetic Algorithms. MIT Press.
10. Haupt Randy and Sue Ellen Haupt (1998). Practical Genetic Algorithms. John Wiley & Sons.
11. 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).
12. P. Adamidis and P. Arapakis, “Evolutionary Algorithms in Lecture Timetabling”, *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*, (1999), pp.1145-1151.
13. 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).
14. 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).
15. E. K. Bruke and J. P. Newall, “Solving Examination Timetabling Problemsthrough Adaptation of Heuristic Orderings”, *Annals of Operations Research,* Vol.129, (2004), pp.107-134.
16. D. Dubois and H. Prade, *Possibility Theory*: *an Approach to Computerized Processing of Uncertainty*, New York, (1988).
17. Z. W. Geem, “School Bus Routing using Harmony Search”, *Proceedings of Genetic and Evolutionary Computation Conference*, Washington, D.C., (2005).
18. 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.
19. 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.

20. 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).
21. 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.
22. 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.
23. C. Reeves, “Genetic Algorithms”, *Modern Heuristic Techniques for Combinatorial Problems*, V. J. Rayward-Smith (Editors), McGraw-Hill International, UK, (1995), pp.151-196.
24. 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.
25. 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.
26. T. Duong and K. Lam, “Combining Constraint Programming and Simulated Annealing on University Exam Timetabling”, *Proceedings of RIVF Conference*, Hanoi, Vietnam, (2004).
27. 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.
28. 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).
29. 10th International Conference of the Practice and Theory of Automated Timetabling Asynchronous Island Model Genetic Algorithm for University Course Timetabling” PATAT 2014, 26-29 August 2014, York, United Kingdom
30. Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2009) “A Guided Search Genetic Algorithm for the University Course Timetabling Problem“ Sadaf Naseem Jat · Shengxiang Yang10-12 August 2009, Dublin, Ireland
31. Int. J. Advance. Soft Comput. Appl., Vol. 2, No. 1, March 2010 ISSN 2074-8523; Copyright © ICSRS Publication, 2010 www.i-csrs.org
32. *International Journal of Innovative Computing, Information and Control* ICIC International “SELF-LEARNING GENETIC ALGORITHM FOR A TIMETABLING PROBLEM WITH FUZZY CONSTRAINTS”2013 ISSN 1349-4198 Volume 9, Number 11, November 2013