University of Sheffield

Timetabling Application for a Small Primary School

Yaqing Yang

Supervisor: Georg Struth

A report submitted in fulfilment of the requirements
for the degree of MSc in Advanced Computer Science

in the

Department of Computer Science

September 13, 2017

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Yaqing Yang

Signature:

Date: September 13, 2017

Abstract

A number of approaches are promoted to solve timetabling problem in the last few years. Timetabling is a complex problem that belongs to NP-complete class problems. The aims and objectives of this project is to develop an automated timetabling application for a small primary school. Focus will be on algorithms and the users interaction. The timetabling system is written by Java. Many academic literatures introduce various algorithms which could be used to construct the school timetable such as genetic algorithms, simulated annealing, tabu search and so on.

The difficulty of this project is to solve the hard constraints and soft constraints that exist in the timetabling problem. And finally, the available result without any hard conflict and most of the soft constraints that I added is obtained. Additionally, users can set the number of class and module distribution of each grade by themselves.

Keywords: primary school, timetable, algorithm, hard constraint, soft constraint

Acknowledgements

First of all, I would like to thank my supervisor Prof Georg Struth, for his guidance and help during the whole process. He guided me when I choose the approach to do my project and make lots of valuable suggestions for my poster and essay.

In addition, I would like to thank my friend Elsa and Rebecca, I kept an active attitude in the completion of my graduation project by their company and support. Especially my friend Yao, he always helped me when I was in trouble. I really appreciate him.

Finally, I would like to thank my family and classmates for their constant support and encouragement, so that I have always had a positive attitude to study and to life. And thank my university: The University of Sheffield, for giving us a good learning environment. I really like library the diamond.

Contents

List of Figures

Chapter 1

Introduction

1.1 Background

There are many approaches to schedule timetables for schools in these years [3]. Timetabling problem is a NP complete problem. Many scholars have done a lot of research in theory, heuristic search technology application solving, expert system application solving and genetic algorithm application. Since 1950s, some people have begun to study the application of computer timetabling in the world. In 1963, Gotlieb put forward the mathematical model of timetable problem in his article [9], which indicates that the study of timetabling has officially crossed the palace of science. The research shows that it is not feasible to solve large-scale scheduling problems only by mathematical methods. After entering the 1990s, the study of timetable scheduling is still very active in the world [2]. At present, using coloring algorithm, simulated annealing algorithm, ant colony algorithm and genetic algorithm to solve the timetabling problem is more common.

Scheduling timetable for schools is a very complicated task that there are two types constraints need to be considered: hard constraints and soft constraints. Hard constraints including one teacher's two classes cannot be arranged at the same time, a classroom cannot arrange two classes at the same time and so on, which are the ones that must to be obeyed. Soft constraints are used to evaluate the quality of the timetable. However, due to that the goal of this project is to arrange a timetable for a small primary school, it does not involve the problems like changing classrooms for each class or considering classrooms capacity. Therefore, the problem is much easier. A certain degree of mobility and interactivity is also required to build the timetable system. There are

many different courses and facilities in the school, creating an effective timetable that includes arranging courses, teachers and classrooms in a specific period to ensure that there is no teacher, course or classroom appear more than once in the same period. Because it is a small primary school, we can assume that the students of each class are fixed and not repeat, which means they have no students in common. In this situation, we assumed a specific combination of teacher, course, class as a set. The same set may be in a week's time but cannot appear more than once in the same period. This is the basic explanation of timetable problem.

1.2 Aims and Objectives

The propose of this project is to develop a reasonable and interactive timetabling application for a small primary school. The timetable not only need to obey the hard constraints: 1) One teacher can only take one lesson at one time. 2) One class can only have one lesson at one time, but also achieve soft constraints as much as possible like the teachers preference for teaching time and so on. Hard constraints are used to ensure the availability of timetable, and soft constraints are used to improve the quality of the timetable.

Focus will be on the algorithms and the user interaction.And Genetic algorithms would be selected to develop the program. Firstly, we conduct the research on genetic algorithm and apply the algorithm to the timetable system. To simulate the course manual work by computer, elements of the problem can be abstracted, constraints could be expressed by mathematical expression. And according to the organization form of timetable and the common law, the scope of the search is reduced of the space in this problem. In this way, course arrangement has been organized effectively, and intelligent is shown in a certain extent.

There are many algorithms could be applied to construct the school timetable: simulated annealing algorithm(SA), ant colony algorithm, memetic algorithms, tabu search, genetic algorithms(GAs) and so on, there are all some applications for all these algorithms. In addition, the method of exhaustion can list all the methods and find the best solution. However, the cost is too high and the time is too long. If there are n periods a week for scheduling courses, and m teachers need to be participated in arranging classes, on average, each teacher is required to attend i class a week. The number of combinations

of courses is nm*i times. Thus, it can be seen, the complexity of the method of exhaustion is very high.

I will select Genetic Algorithms to do my project at first. Because I read some articles said GAs could provide a set of different timetables, which is more flexible for the users. If some problems encountered which cannot be solved, I would use the other algorithms like simulated annealing algorithms to perfect my project.

There are several kinds of programming language could be selected too. I tried to use objective-c at first, however, I failed in solving the return type problem about array list. Thus, I back using java as my programming language.

1.3 Contributions

During the project, firstly I learned the difficulty of solving the problem of the timetable is mainly to solve the problems of hard constraints and soft constraints. The hard constraints are the ones that the timetable must obey, which is the basic principle for checking the availability of a course schedule. The soft constraint is added to make the timetable more practical and interactive, and improve the quality of the curriculum.

After studying several algorithms that can be used to solve timetabling problems, I chose the genetic algorithm to solve the problem. Because compared to other algorithms, genetic algorithms can get a lot of available solutions in a shorter time. When applying genetic algorithms to timetabling problem, the first difficulty is to present the lessons of one week for one class by a gene, and genes of all classes form a single chromosome that represents the school timetable. Then in the calculation process, generation was selected by tournament selection method. The second difficulty is the application of crossover and mutation to chromosomes. The most important thing I think, is the calculation of the fitness of each chromosome, which combine chromosomes with various constraints. I think I've found a good approach to calculate the fitness of each chromosome and get the one with best fitness, which is the final result.

Finally, I managed to use genetic algorithm to solve the timetabling problem and put forward a practical method for calculating the fitness of each generation, and get the timetable which resolve all hard constraints and some soft constraints. Whats more,

the other soft constraints can be added in the code manually, and the timetable can be scheduled according to the importance of each soft constraint.

1.4 Overview of the Report

The second chapter mainly introduces the research status and background of the project, I will introduce relevant materials I surveyed, including several different approaches of arranging the school's timetable issues to demonstrate my awareness and understanding of the background literature to the projects topic.

The third chapter discusses and analyses the basic requirements of the timetabling problem which should be satisfied according to the principles and types of timetabling problems, and break the project down into manageable steps.

The forth chapter details the design technique chosen from the many available algorithms. System basic functions are firstly introduced and then describes the algorithm model when applying the algorithm to timetabling problem. Then introduces the interface design of the system.

The fifth chapter illustrates how I apply genetic algorithm to the timetable system using java, details the code structure I wrote, and how I implement the users interface. And I will give a self-evaluation in this chapter, which will explain the time I cost to learn basic knowledge and the hard part and difficulties I met and how I solve them.

The sixth chapter shows all the results of my work, the calculation process and how the whole system work. In addition, I will discuss the degree of the goals achieved and further work which could make my system better.

The seventh chapter summarizes the whole paper: my work and the evaluation, and explains if I had to do this again, how can I make the system better and what improvements I could make.

Chapter 2

Literature Review

2.1 What is the Timetabling Problem

Course scheduling is a very complex and important work of the school. Its essence is for schools to set an appropriate set of teaching practice and space for courses offered. In this way, the whole teaching plan could be carried on orderly [19]. The timetabling problem is a NP complete problem, which integrates the dual constraints of time and space. Mathematically, the timetabling problem is a combinatorial programming problem in the four dimensions of time, teacher, student and classroom, with teaching plan and various special requirements as constraints [17]. (In this dissertation, since it is a primary school, it is assumed that the students and the classroom are fixed. Thus, it is only possible to consider the three-dimensional space: time, teacher and the classroom.) The essence of this problem is to solve the conflicts between the factors. The timetable which have been obtained as result should ensure that the teachers scheduled in the timetable are not in conflict with the classroom, which means, only one teacher can attend one class in a period. And the course schedule should meet the requirements of teachers and classroom resources constraints as much as possible.

When creating scheduling algorithms, there are two kinds of constraints need to be considered: hard constraints and soft constraints [12]. Some of the typical hard constraints are:

- one teacher can only take one class at a time
- one classroom can only have one class at a time
- classroom capacity should be greater than the number of students of the class

- the classroom is equipped with all teaching equipment

Soft constraints may be the teacher's time / class preferences, try to arrange the required courses in the morning rather than in the afternoon and so on.

In this problem, hard constraints are the ones that must be observed. However, multiple soft constraints may have conflicts and need to be weighed between them. For example, in order to achieve better teaching quality, the math class to be scheduled in the morning will be given priority to a teacher who likes to take classes in the afternoon.

The iterative method can be used to find the optimal solution of the timetabling problem. However, as the amount of data increases, when the possible number of class configuration makes the iterative method infeasible, the optimization algorithm model can be used to find the optimal solution in a reasonable time [12].

2.2 Approaches to Solve Timetabling Problem

2.2.1 Simulated Annealing Algorithm

Theory and idea

The simulated annealing algorithm is derived from the principle of solid annealing, heating the solid to a high temperature, then let it cool slowly. In the process of heating, the particles inside the solid become disorderly with the increase of temperature, and the internal system energy can be increased as well. When cooled slowly, the particles gradually become orderly, reach equilibrium state at each temperature, and finally reach the ground state at room temperature, and the inner energy can be reduced to the minimum. The combination of solid simulation is used to solve the optimization problem. The internal energy is simulated as an objective function, and the temperature is evolved into a control parameter. Then a simulated annealing algorithm is proposed to solve the combinatorial optimization problem. The initial solution is repeated to generate the new solution, objective function is used to calculate difference between the two solutions, accept or abandon the iteration, and gradually decrease the temperature value. When the algorithm terminates, the solution obtained is the relatively optimal solution, which is a heuristic random search method based on Monte Carlo iterative process [18].

Basic idea:

(1) Initialize the initial temperature T, take the state with maximum T as the initial state, and the iteration number of each T is L;

(2) For k=1, 2, , L, do steps (3) to (6);

(3) Produce the new solution S;

(4) Calculate the variable quantity in the evaluation function of the two states;

(5) If the change value is negative, the new solution S is used as the current solution, otherwise S is accepted as the current solution by a probability.

(6) The termination conditions are usually set as when several successive new solutions are not accepted, the algorithm is terminated.

(7) When the temperature gradually decreases to 0, returning the second step.

Implementation

To solve the timetabling problem with this algorithm, course initial coefficient of scheduling difficulty need to be set according to the amount of data firstly. Then calculate the weight of each factor in the system according to the relationship between the elements of classrooms and teachers in the practical, and calculate the scheduling difficulty for each data, that is to initialize the data to obtained the initial temperature T.

Based on the weight of each data, finding the maximum weight data at first, which is the most difficult to schedule. From this data, other requirements can be solved and the first new solution could be got. After scheduling this data successfully, the difficulty coefficient of each element is calculated immediately. Calculating the difficulty coefficient according to the operation method of the first step for data which has not been scheduled yet, calculate the change in temperature as the new evaluation standard for data, which is the fourth step above.

Repeat the operations above until the latest temperature changes of all data are 0, and the output schedule is obtained by optimizing the row data.

2.2.2 Ant Colony Algorithm

Theory and idea

There are many kinds of social insects in nature. Although their individual behaviour is extremely simple, complex collective behaviour in an orderly manner are exhibited by the composed of individual groups such as finding the shortest path between the food and the nest. Through observational studies, bionic scientists found that ants were able to find the shortest path between nests and food, based on the substance they release as pheromone. When ants crawl on the ground, they release pheromones on the path. During ants moving, when there is a bifurcation without any clue, and from the point of view of probability, there would be half ants go down and the other half ants go up. As shown in figure 2-1.

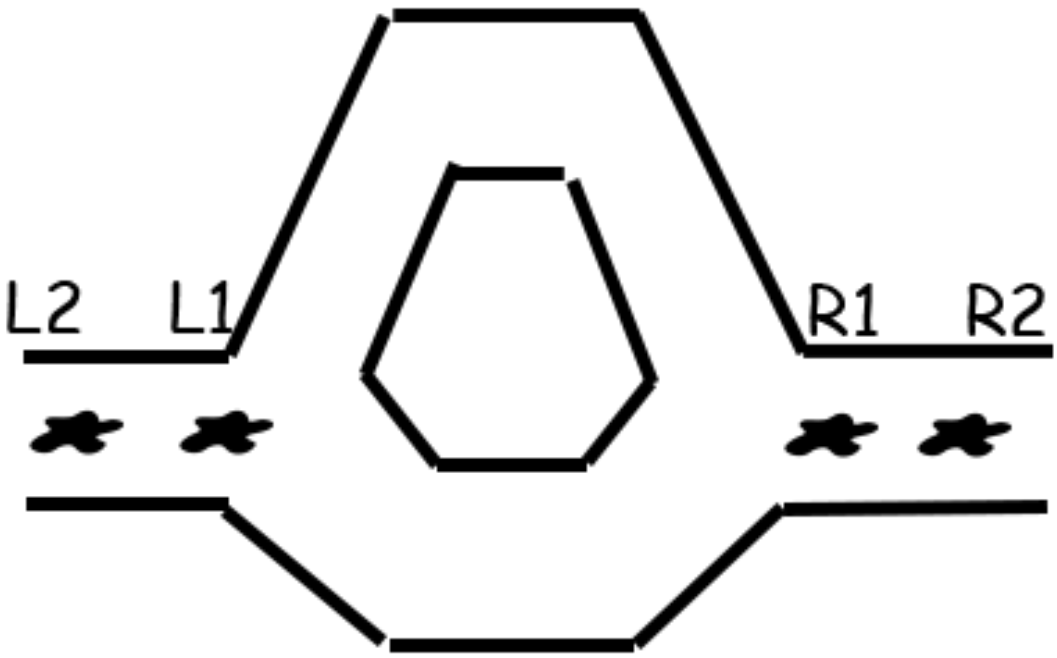

Figure 2.1: Ant paths' choices

All the ants are assumed with the same speed, then the ants on the short path will arrive first obviously. When L1, R2 arrived at the fork in the road, L2, R1 are on the top path, so that the pheromone is released more in the bottom path, concentration growth is also faster. And the ant is selecting a path according to the concentration of

pheromone, which provides the path information for later ant as well.

As a result, even if there are more paths available to the ants, they can still find the shortest path. Such collective behavior of ants shows a positive feedback phenomenon. Through the exchange of information, ants achieve the goal of collective search for food. Ant colony algorithm is the simulation of such an optimization mechanism, that is, through the exchange of information and collaboration among individuals, finally find the optimal solution. This is a new type of simulated evolutionary algorithm in recent years [4].

Implementation

Ants in nature exist in a three-dimensional environment, however the problem space is usually solved in an abstract plane. Since the route ants take for seeking food is a straight line, which can be abstracted in a two-dimensional space. However, since the computer deals with discrete events, it is impossible to describe a continuous plane completely. Thus, as long as the continuous plane is discretized into a discrete plane composed of a set of points, the ants move through one point after another. And when the ants complete a move, it will carry out the evaporation of a trajectory. The solution space of the problem can be described by graph, and then the ant colony algorithm has the possibility to be solved.

The search time of ant colony algorithm is too long. Moreover, when running a certain number of times, it is easily appearing large numbers of ants gathering in several paths because of continuously improved optimal solution, which would lead to stagnation and can only get the local optimal solution. In order to overcome these shortcomings, many scholars have proposed improved algorithms, such as meeting algorithms [10]. Ant colony algorithm has a better application in combinatorial optimization problems, such as traveling salesman and job scheduling problem. In the research of ant colony algorithm, most of them are theoretical research and simulation experiments based on the traveling salesman problem [16].

2.2.3 Memetic Algorithms

Theory and idea

Memetic Algorithm (MA) is an optimization algorithm put forward by Pablo Moscato [15] based on simulated cultural evolution. Corresponding to the simulation of biological evolution of the genetic algorithm, the propagation process of cultural genes should be strictly replicated. If there exist mutation, the mutation of each step needs a lot of professional knowledge to support. In addition, all mutations should bring progress and not chaos. That is the reason of the speed of cultural evolution is much faster than the evolution of biologicalevolution. The cultural genetic algorithm uses local heuristic search to simulate the process of mutation that supported by a large number of specialized knowledge. Thus, it is essentially a combination of population-based global search and individual-based local heuristic search. Memetic algorithm is a kind of extension of the genetic algorithm which the basic tuples are replaced by memes that can be raised itself in the lifecycle, unlike genes. The University of Nottingham applied this algorithm to the timetable of the study [1]. Mutation operator, which is applied to select parents from the population, would disrupt each timetable. This combination will produce new solutions with better quality.

Implementation

For a given optimization problem, a certain number of initial individuals can be determined at first. The states of these individuals can be random or be determined by a heuristic mechanism. Then each individual is searched locally, and the fitness of the individual is improved by local search. After the population reaches a certain reserve, the interaction operation between individuals can be carried out. Such interactions can be mutually competitive or mutually cooperative. Competing operations are similar to the individual selection process in genetic algorithms. The cooperative behaviour can be regarded as the crossover mechanism of genetic algorithms or other methods of generating new individuals, and can also be generalized as the process of exchanging information. Local search, competition, and cooperative operations are performed in loops until the termination condition is satisfied [13].

2.2.4 Tabu search

Theory and idea

Tabu search method can only obtain one viable schedule just like the simulated annealing algorithm. Tabu Search (TS) is another evolutionary heuristic that updates a single solution. Originally proposed in [7] [8]. The idea behind TS is to start from a random solution and move it continuously to one of its current neighbours. Each time the move is performed, the opposite is a prohibited list of prohibited fixed-length lists. From a given solution, usually cannot reach all the neighbours. In fact, a new candidate moves to bring the solution to the best neighbour, but if the move exists in the tabu list, it is only accepted if the target function value is reduced to the desire level (which is the lowest level) that the target reaches.

Implementation

The algorithm they implement includes a variable-sized tabu list: specifies the minimum and maximum lengths, and during the search process, the actual length is changed randomly after the number of iterations. The tabu search algorithm we use is very similar to the algorithm proposed by Hertz [11]. We implemented a variable-length taboo list and used the objective function to evaluate the cost of the schedule, such as the SA. However, the results obtained using this simple model are not satisfactory, in this way we have achieved the same relaxation procedure as SA. The results are indeed very good and reliable to produce a lower cost than SA and GA.

2.2.5 Genetic Algorithms (GAs)

Theory and idea

Genetic algorithms (GAs) is a computational model that simulating the evolution of Darwin's genetic selection and natural selection. It is a method of searching for the optimal solution by simulating the natural evolution process. Genetic algorithms(GAs) is often used to optimize the solutions [14]. Unlike other heuristic schemes produce only one sub-optimization method each time, GAs will retain a lot of independent solutions

to form a population. Parents are selected from this generation of population to form a new individual as their child. These individuals will inherit properties from their parents which may be indicated by specific genes. If these characteristics of the child make him more adaptable to the environment, then the probability of his survival will increase. At the same time these features will be inherited to the next generation of population by genes. On the contrary, if the child has some characteristics make him cannot adapt to the environment, then the child will die before the next generation, and these features will not be inherited. This is the process of survival of the fittest. Because genetic mutations maintain the diversity of species, they also create newer and better individuals. To look for an optimal solution from a great deal of solutions is the principle of genetic algorithms. The generation is composed of many different timetables, and many of them are not optimized. However, select parents from these timetables and combine them with excellent genes, we can obtain a better timetable.

Implementation

Using genetic algorithm to solve optimization problem, the first step is to encode the points in the feasible domain (generally use binary encoding). Then, in the feasible domain, some coding groups are selected randomly as the starting point of the evolutionary origin, and the objective function value of each solution is calculated, that is, the coding fitness. Then, just as in nature, the selection mechanism is used to randomly select code from the coding group as a coding sample before the propagation process. The selection mechanism should ensure that the solutions with higher fitness can retain more samples, while the lower ones retain fewer samples and even be eliminated. In the next steps of reproduction, the genetic algorithm provides two operators: crossover and mutation, to exchange the selected samples.

The crossover operator switches two bits of randomly selected code, while the mutation operator directly reverses one of the randomly chosen bits in the code. The next generation of coding groups is generated by selection and reproduction. Repeat the selection and propagation process until the termination conditions are satisfied. The optimal solution in the last generation of evolutionary process is the result obtained by using genetic algorithm to solve optimization problems.

The general flow of solving combinatorial optimization problems using genetic algorithms is described in mathematical language as follows:

1) Encoding the relevant parameters of the problem to be solved;

2) Randomly initialize populations X (0) = (X1, X2, ..., Xm), where m is the group size;

3) The fitness value f(xi) is calculated for individual xi in the current generation X(t), and the fitness value reflects the performance of the individual;

4) The selection operator is used to generate the intermediate generation X (t);

5) A new generation of population X (t+l) is generated by applying crossover and mutation operators to X (t). The purpose of crossover and mutation operators is to extend the coverage of finite individuals and embody the idea of global search; f evolved to the next generation, generation counter t plus 1. If the termination condition is satisfied, the next step is performed, otherwise, back to step 3);

6) Output approximate optimal solution.

The corresponding execution block diagram is shown in figure 2-2.

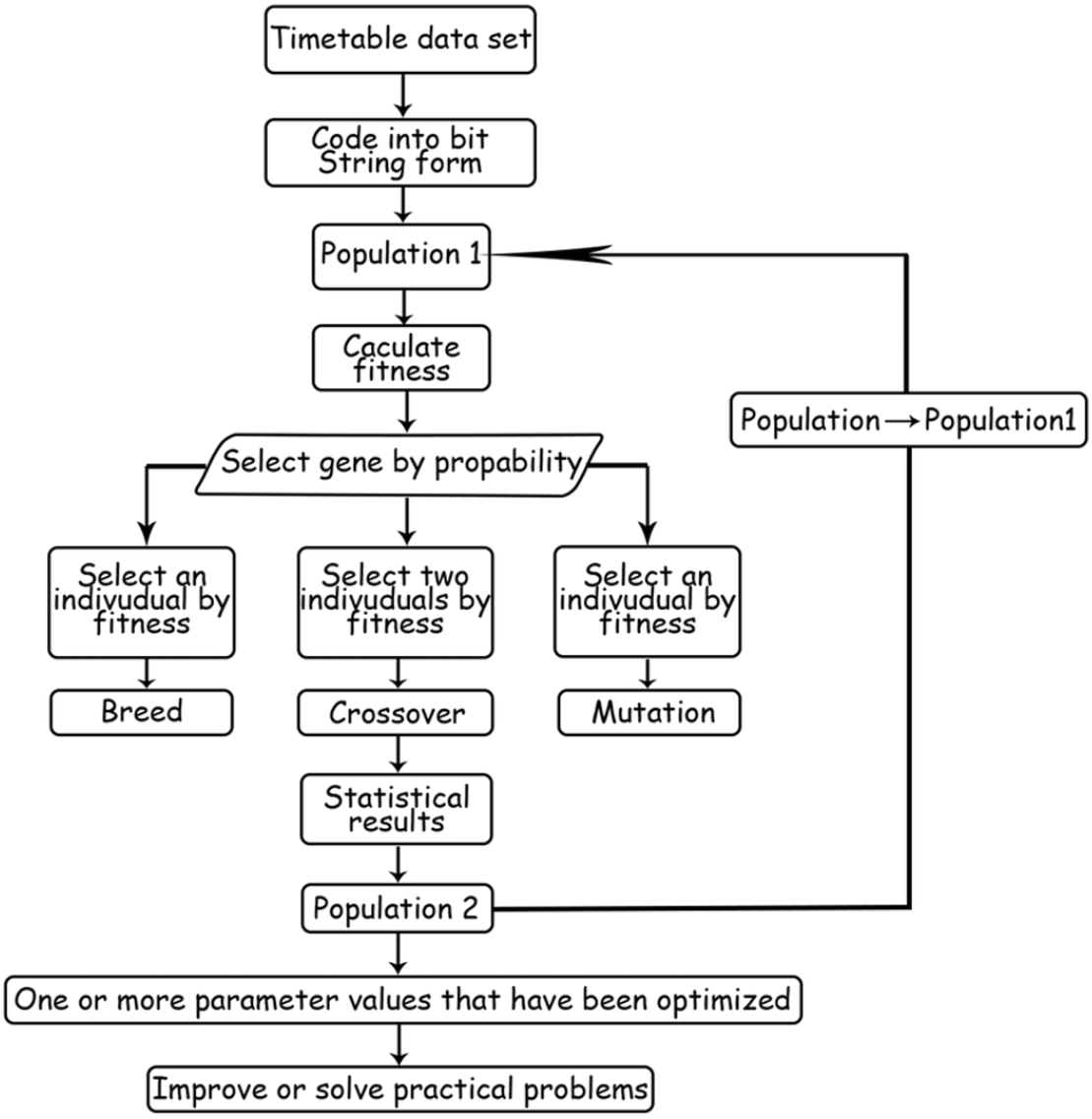

Figure 2.2: Genetic Algorithms diagram

2.2.6 Greedy Algorithm

Theory and idea

Greedy algorithm is an improved hierarchical processing method, and the optimal solution is constructed step by step. It is from a certain initial solution of the problem, make a series of greedy choice under the certain standard (choice once made, it cannot be changed), that seems to be the best choice of current state. Gradually approaching

the given target, and as quickly as possible to obtain a better solution [18]. When a certain step in the algorithm is no longer allowed to go ahead, then it stops. The core of greedy algorithm is to select a maximum weight optimal strategy as the current strategy in the chosen strategy. Therefore, the quality of greedy algorithm is mainly determined by the weight.

Implementation

In the course scheduling system, the greedy algorithm is starting from an initial state of timetabling problem. On the basis of the greedy strategy to make a step forward towards the eventual goal: schedule all the courses, determine whether we can find a solution of elements of feasible solutions. If the above is feasible, continue to proceed to the given target according to the greedy strategy, and find the next solution element until progression can no longer continue. Finally, a feasible solution of the problem is formed by all the solution elements obtained. At this point the algorithm is over.

The disadvantage of greedy algorithm is that the solution is not very good, while the biggest advantage is the extremely low complexity of time. Greedy algorithm can get some local optimization solution in a sense. It cannot be back, can have the aftereffect. Normally it does not satisfy the principle of optimality and it is not applicable to solve the feasibility problem, but only applicable to the optimization problems which are easier to obtain feasible solutions. In order to minimize the side effects of the greedy algorithm, to make the final solution closer to the optimal solution, the efficient optimization algorithm, such as dynamic programming could be used as much as possible in the algorithm. Greedy algorithm can also provide a better initial boundary value for the search algorithm.

2.2.7 BacktrackingAlgorithm

Theory and idea

Backtracking algorithm, also known as heuristics, it is a method of systematically solving the search problem, which can be considered as a pruned DFS (depth first search) process. It searches forward according to the priority condition to achieve the

goal. However, when searching for a certain step, it is found that the original choice is not superior or cannot reach the goal, then turn back and choose again. A state point that satisfies the backtracking condition is called a backtracking point. Specific to the computer intelligent course arrangement system, the optimization condition is the constraint group in the mathematical model of arranging courses. The mathematical relation formed by the interaction of the characteristics of the elements in the demand and the element characteristics of the resources. If the constraint group is not satisfied, the choice is either not optimal or not up to the target. When a state traverses all the possible steps but fails to satisfy the constraint group, the state satisfies the backtracking condition, which is the backtracking point.

Implementation

When using backtracking algorithm to solve the timetabling problem, the form of the solution should be described and a solution space should be defined which contains all the solutions of the problem. Secondly, the state space tree needs to be constructed, and each path of the tree represents the possibility of a solution. Thirdly, the structure constraint function is used to remove the illegal solution by describing the general features of the legitimate solution, so as to avoid searching for the rest part of the illegal solution. backtracking is then finished by depth first search [20].

1) Set the initialization scheme (give the variable initial value, read into the known data, etc.);

2) Change the way to test. If all the test done turn to 7);

3) Determine whether the method is successful (through the constraint function), if unsuccessful, turn to 2);

4) If test success, then move forward and try again;

5) If the correct plan has not been found, turn to 2);

6) A solution has been found, then record and print it;

7) Step back (backdate), if not to turn the root, then turn to 2);

8) the root node has been dropped, then the course is finished or printed without timetable results.

Backtracking algorithm applies to a class of problems that are quite large but still finite in solution. An important feature of it is that the search and results produced are implemented at the same time. At any point during the search, only the path from the start node to the current node is retained. Thus, the spatial requirement of the backtracking algorithm is a constant, that is, the length of the longest path from the start node. This property count a great deal because the size of the solution space is usually the exponent or factorial of the longest path length. Thus, if all the solution space need to be stored, there is not enough space anyway. The drawback of backtracking algorithm is that the time complexity is large, so caution is needed in the adoption. Using it in conjunction with other algorithms would be better.

Chapter 3

Requirements and Analysis

This chapter discusses and analyses the basic requirements of the timetabling problem which should be satisfied according to the principles and types of timetabling problems. The aims and objectives of the project is to construct a reasonable and interactive timetable for a small primary school. Focus will be on the algorithms and the users interaction.Algorithms used would be selected from the second chapter. The users interaction means it is not a simple, fixed timetable. This timetable could be edit by the administrator by selecting modules of each grade, entering the number of classes and adding soft constraints.

3.1 Principles and Requirements

Complex timetabling problems exist in five-dimension relationship: classroom, module, class, time, and teacher. But since it is a primary school, the constraint of changing the classroom between each lesson is not considered, in other words, each class has a fixed classroom and will not change the location. Therefore, there is only a relationship between teacher, curriculum, class, and time in this timetabling problem. Hence many hard and soft constraints need to be considered when creating scheduling algorithms. There are some typical hard constraints in the timetabling problem:

- The same teacher can only teach one course at the same time
- The same class can only take one course at the same time (which is not need to be considered in this project)
- The classroom must contain all the equipment needed (which is not need to be considered in this project)

The above constraints are the most basic requirements of the timetable system which must be avoided in the course scheduling system. Once it happens in the course arranging system, it will inevitably affect the teaching task and teaching quality.

A course schedule that meets all hard constraints is a qualified curriculum timetable. But in order to improve the quality of the timetable and achieve the best teaching effect, there are some constraints that should be implemented as much as possible. These constraints are called soft constraints. For instance:

- Course time balance: for example, math or English class should be arranged in the morning and P.E is better to be scheduled in the afternoon.
- Teacher's time preference: such as a teacher does not like to teach in the morning every day.
- Teaching time balance: try no not schedule one teacher for several continuous curriculums in one day.

3.2 Typical Subject

Timetabling problem is a multi-index optimization decision problem. It involves many factors and its structure is very complicated, it is also a typical problem in combinatorial planning. Next is a brief analysis of the timetabling problem:

Suppose the primary school has C classes, L courses, T teachers and S time periods.

- Set C= All the classes
- Set L= All the lessons
- Set T = T1, T2, Tm is the set of all the teachers
- Set P= All the teaching time of each week

Thus, the problem of arranging courses can be summed up as: $f : L \to C * P$. That is, each course corresponds to a suitable classroom and time. But this description is incomplete and does not meet the requirements of the previous section.
Suppose Tl indicates the class time of the lesson, among which $l \in L$, $T_i \subseteq T$.
Lt stands for all the courses that teacher t teaches, among which $t \in T$, $L_t \subseteq L$.

Lc represents all the courses class c will takeamong which $c \in C$, $L_c \subseteq L$.
Assume that $CConflict = \sum^{i \in C} conflict(i) = \sum^{i \in C} \sum^{j,k \in L} < T_j, T_k >$, $CConflict = \sum^{i \in T} conflict(i) = \sum^{i \in T} \sum^{j,k \in L} < T_j, T_k >$

The timetabling problem can be formulated as follows: $f : L \rightarrow C * P$.

Constraint condition, (totalamountrestriction)

The solution f, which satisfies all the above three constraints, is a feasible solution to the timetabling problem. Intuitively, the solution is not unique and should have multiple solutions. Moreover, the model only considers the hard constraints and does not consider any soft constraints. Of course, a high-quality curriculum should be multi objective optimization, that is, to pursue optimal allocation on multiple resources (i.e., consider more soft constraints as much as possible). However, in fact, it is impossible to find an optimal solution. In practice, a relatively optimal feasible solution is always obtained. Therefore, we should abandon the attempt to get the absolute optimal plan. In addition to hard constraints, the schedule can be considered feasible as long as it meets the reasonable and practical requirements of manual scheduling. Finding the relative optimal solution in all feasible solutions is one of our goals. A complete schedule should contain the information combination of all grades, classes and courses. If the relative optimal solution is found focus on a single course, it is not necessarily a good solution to put in the full timetable, which may lead to the difficulty and failure of other courses.

3.3 Project Restriction

Timetabling Problem is a NP-Completed Problem. From the 60s and 70s of last century, the computer industry has done a great deal of theoretical research and exploration on the arrangement of computer-aided curriculum. In the middle of the 70s, some scholars made outstanding contributions to this issue. S.Eve et al published a paper in SIAMJ.COMPUTR magazine for the first time to demonstrate that the timetabling problem was NP-Completed, and that the timetable problem was theorized [5]. At present, there is no polynomial algorithm to solve the NP-Completed problem. It means that it is impossible to find the solution of the timetabling problem with a polynomial [6]. Moreover, the theory and practice show that, under the premise

of meeting various requirements and restrictions, as long as the information involved in curriculum schedule changes slightly, it may lead to drastic changes in schedule selection programs. Although the current hardware development is relatively fast, but since the algorithm itself has no breakthrough, it is difficult to find the suitable solution. However, according to the specific characteristics of the timetable system itself, there are still some good ways to give some higher quality solutions to the problem.

3.4 Suitable Approach

Genetic algorithm based timetabling system can be implemented by a variety of code languages, such as c++ and java. However, since the familiar to the Objective-C, I first tried to write with objective C, but because there is no way to solve the problem that make array directly as the return type and passed as parameters, finally Java was chosen to development this application instead.

3.5 Evaluation Techniques

Whether the result of the timetable system meets all the hard constraints, whether there is conflict or not, if there is no conflict, then it is the feasible result. After the feasible results are obtained, what soft constraints are satisfied by the final results obtained by the timetabling system could be used to evaluate the quality of the schedule obtained.

Chapter 4

Design

4.1 System Basic Functions

4.1.1 Input and maintenance of basic data

The basic information needed in the course arrangement of primary school, such as teacher, class, class time and so on, is input into the corresponding data file, and the information can be queried, modified and deleted. Since the timetable is designed for small primary school with less data, we consider only manually input data.

4.1.2 Generate timetable automatically

After entering all the relevant data, the system arranges the time of the courses and assigns the right classroom automatically. All courses will be arranged at one time during the scheduled course time. This is also the core function of the system.

4.1.3 Modify constraints

In the case of satisfy all hard constraints, we can modify the soft constraints to re schedule the timetable. For example, adjusting the math class time, trying to satisfy the individual teacher's preferences, adjusting the class time for them, etc., to make the quality of the timetable produced by the system higher.

4.1.4 Interface design

Design an interface which could set the number of class of each grade and the present the timetable, to make the interface have better interactivity, and make users can see the results more clear and intuitive.

4.2 Algorithmic model

The genetic algorithm is used to solve the optimization problem. Firstly, the point in the feasible domain is encoded, and the binary encoding is usually adopted. Then, in the feasible domain, some coding groups are selected randomly as the starting point of the evolutionary origin, and the objective function value of each solution is calculated, which is the coding fitness. Then, just as in nature, the selection mechanism is used to randomly select code from the coding group as a coding sample before the propagation process. The selection mechanism should ensure that the solutions with higher fitness can retain more samples, while the lower ones retain fewer samples and even be eliminated. In the next steps of reproduction, the genetic algorithm provides two operators, crossover and mutation, to exchange the selected samples. The crossover operator switches two bits of code selected randomly, and the mutation operator directly reverses one of the randomly chosen bits in the code. Thus, the next generation of coding groups is generated by selection and reproduction. Repeat the selection and propagation process until the end conditions are met. The optimal solution in the last generation of evolutionary process is the result obtained by using genetic algorithm to solve optimization problems.

According to the characteristics of the class arrangement in the primary school: the classroom is relatively fixed and the teaching time is relatively fixed, we make the following assumptions for the system:

1) The classrooms of each class are fixed. One class only takes lessons in one place.

2) One lesson has only one teacher at the same time.

3) A teacher is not allowed to attend classes in many classrooms at the same time (such as self-study sessions).

4.2.1 Encoding

The encoding method used in the timetable scheduling program in order to effectively encode all the class attributes required: the timing of the course, the teacher and the class, a number could be assigned to the time, the teacher and the class. Then, in a familiar way, integer arrays are used to encode chromosomes. This means that each scheduled class needs only three integers to encode, as shown in the figure:

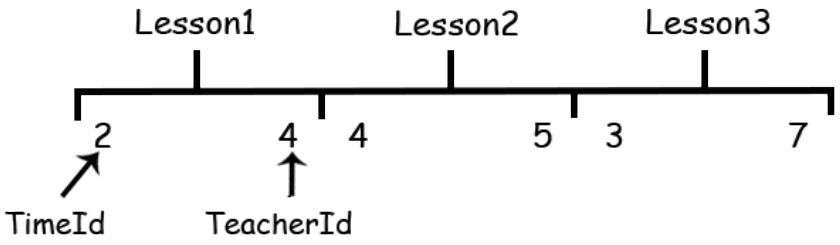

Figure 4.1: Course arrangement encoding

One week of classes is made up of one chromosome. One of the genes in a chromosome is the timetable of a class.

Firstly, create a container class for each type data (classroom, teacher, time and module). Secondly, build a Lesson class that represents all of the above information: a class at a given time, a particular teacher, and a specific course. Then build a Timetable class to get all available classrooms, time and a teacher, read the chromosomes, create a subset of the class from the chromosome, help to assess the fitness of the chromosomes, and calculate the conflicts.

4.2.2 Crossover

Firstly, a new empty population is created for the next generation, then the population is traversed, and the crossover rate is used to consider the crossover of individual. If the individual does not cross, it directly joins the next population, otherwise a new individual is created. The method of filling chromosomes in offspring is to traverse the parent chromosomes and then randomly select genes to join the chromosomes

of offspring. For individual of the population to complete the crossover process, the crossover method returns to the next generation of population. Thus, a single crossover is enough to evolve populations. However, if there is no mutation, the genetic algorithm can easily fall into the situation where the global optimum cannot be found and only the local optimum can be found.

4.2.3 Mutation

Genetic algorithm selection variability and crossover techniques are often determined by chromosome constraints. In this project, chromosomes are made up of teachers and classrooms at specific times, and random numbers cannot be simply chosen. Moreover time, the teacher and classroom ID have different range as well, cannot choose random number simply. If ID is assumed to be continuous, then we can select random numbers for each different type of object: teacher, time and classroom, however, in fact their ID may not be continuous [21].

In uniform crossover, genes randomly selected from existing, effective parents may not be the best, but effective individuals. Mutations can also be implemented in a similar way. A new, random, but effective individual was created, and then randomly selected from the individual gene until the mutant individual was obtained. This technique ensures that the mutated individuals are fully effective and only select meaningful genes, which called "homogeneous variation".

Population variation is achieved by looping the non-elite individuals of the population, generating a random but effective individual and randomly copying genes from it.

4.2.4 Fitness calculation

The initial population is established, these individuals need to be evaluated and calculated for their fitness values. In timetabling problem. The objective of optimizing timetable is to obey some soft constraints without breaking the hard constraints. If there is no soft constraint, the fitness of any two valid solutions returned by the scheduling procedure is not different, so the soft constraint determines the quality of the timetable. Using a method calcFitness function to calculate the fitness, let fitness = 1/(conflicts+1), and output the fitness. This expression means that the fitness value of an individual is inversely proportional to the number of constraints that has been violated.

The key to add soft constraints is to reconcile the important relationship between hard constraints and soft constraints. If a soft constraint is assumed to be one "conflict" as well, it is equivalent to a hard constraint. In this case, the model may be given an invalid solution because it satisfy some soft constraints to compensate the fitness broke by the hard constraints.

Thus, when calculating the fitness by function calcConflictsa new scoring system is considered

- If hard constraints are violated, 1000 conflicts are added, such as classroom conflicts and teacher conflicts.
- On the contrary, if the violation of the soft constraint is violated, 1-3 conflicts are added according to the severity of the constraint, in order to show the necessity of compliance with the hard conflict, and the priority of the soft constraint can also be arranged. The method returns the total number of conflicts the system finds.

A calcFitness method is then used to calculate fitness values: fitness=1000/(conflicts+1000) and output the fitness obtained.

The fitness value for the perfect solution should be 1, adding this condition to the GA class as the second termination check condition (the first type is the number of generation, such as setting the 1000 generation as restriction evolution limitation).

- If there are 0 conflicts in the system, fitness is 1, and there is no change.
- If There is no hard constraint in the result and it is valid.
- If , there exists hard conflicts, and the resulting solution is invalid.

4.3 Interface design

The system has been designed as three interfaces, the first one is the welcome interface, which is used to present some basic tips and allow users to set the number of classes per grade, and set a button to jump to the next interface.

The second interface is used to set the module for each grade. The user can set two categories content: modules type in each grade, and the number of courses per week. Among them, the total number of module hours per class per week is 15, otherwise when the button is pressed there will be issued warning. Click the button, if the calculated results still exist hard conflict, then pop-up a warning as well, and the system cannot jump to the next interface. Otherwise, jump to the next interface to display the results.

The third interface is the interface that displays the timetable, Timetables of 6 grades are displayed through 6 JPanel using CardLayout, and they can be jumped to by clicking the 6 buttons. The timetables of each grade are also arranged according to classId and time.

Chapter 5

Implementation and Testing

5.1 Programming Language and Platform

Java is chosen to implement the whole application although there are several kinds of languages to choose from. Java is one of the most widely used network programming languages at present. It is simple, object-oriented, stable, platform independent, explanatory, multi-threaded, dynamic and so on. And Eclipse is selected as the development environment. Eclipse is a well-known cross platform integrated development environment (IDE). It was originally developed for Java language development, but it has also been developed by adding some plug-in development tools for other computer languages such as C++ and Python. Eclipse itself is only a framework platform, but the support of many plug-ins makes it difficult for Eclipse to have other functions and relatively fixed IDE software. Many software developers develop their own Eclipse as a framework for IDE.

The main features of Eclipse are as follows:

1) An open, extensible IDE: Eclipse platform is the basis for building blocks and constructs and for running integrated software development tools. Eclipse platform allows tool builders to develop tools seamlessly integrated with other tools independently, where users cannot even distinguish where a tool function ends, and where another tool function begin.

2) A successful underlying graphical interface: The swt development kit, written by Eclipse, provides a better choice for Java programmers beyond awt and swing. SWT itself is only a set of underlying graphical interfaces API written by Eclipse

organizations for developing Eclipse integrated development environments.

3) Powerful plug-in loading function: plug-ins can be added continuously. At the same time, plug-ins can be added to the existing plug-in as well, and then realize the expansion of the function. Currently, Eclipse has begun to provide functional plug-ins for C language development.

4) Version management control can be implemented cheaply by importing some CVS related plug-ins: The Eclipse platform provides support for team development operations directly from the workspace. This support allows developers to interact with several independent repositories and different versions of code or projects concurrently.

5.2 Implementation

5.2.1 At the beginning

First of all, some data need to be added to the timetabling system. Specifically, there are classrooms, teachers, time and grades, and then build a timetable around these data.

First, create a container class for each data type to define the basic properties of each data type.

1) Class Classroomstore information of each class. The information includes class number, module id which would be taken in this class and the grade.

2) Class Timesave the time information for each class. The corresponding attributes are the ID and date (e.g. Fri 9:00-10:30).

3) Class Teacherstore the information of teachersthe corresponding attributes are the teachers ID and name.

4) Class Module this class is to store information about one lessoncorresponding attributes are moduleId, moduleName and the TeacherId corresponding to this course. A method to select teacher randomly is provided in the class.

5) Class Lesson, this class is used to represent a class of a particular grade which is having a particular course at a given time by a particular teacher. It is a combination of the all above information.

6) Class Timetable, to encapsulate all objects. In this class, constraints and interactions between objects are implemented, and classes, time, teachers, and courses are added to the timetable. In this way this class achieves two goals, one is the Timetable object knows all of the available classes, teachers, time, and the Timetable object can read chromosomes, and create a subset of the class to assess the fitness of chromosome. There are two important methods in this class: createLessons and calcConflicts.

   - The method createLessons get an individual chromosome at first, thus get the total class number and the total number of courses, then read the chromosomes, and put the time, the teacher, assigned to each of these classes. Therefore, the method ensures that all courses and classes are taken into consideration.
   - The method calcConflicts is used to count the number of conflicts. In case of any violation of any hard constraint, the amount of conflict will be increased by 1000. Hard constraints include a class that has only one class at a particular time and a teacher who has only one class at a particular time. In addition, if the violation of the soft constraint is considered, the importance of the constraint and the number of conflicts are considered. I added two soft constraints which could be modified anytime if needed, one is math or English class should be scheduled as far as possible in the morning, if the violation is conflict plus 1. The other one is teacher Yao does not like teaching on Monday morning or Friday afternoon, if the violation is in violation, the number of conflicts will add 3. The importance of the hard constraint is controlled by the ratio of violating the hard constraint and violating the soft constraint. Because if there are still hard constraints, the timetable will not be available.

7) Class Ga represents the genetic algorithm itself, and provides the interface method for the realization of the problem: cross, mutation, fitness calculation and termination condition checking.

8) Class Individual represents a candidate solution that is primarily responsible for storing and operating a chromosome.

9) Class Population provides the needed functionality to manage a set of individuals in a group. An array made up of individual is preserved, and the method in the individual can be inquired and updated, the overall fitness of the population is stored as well.

5.2.2 Interface

1 Welcome interface

I chose to use GUI to write the interface, first of all, the welcome interface. This interface is written in class MainJFrame.

In this JFrame, three JPanel are added. On the first JPanel, a JLabel is added. And add a JLabel and a JTextField on the second JPanel. On the third JPanel, adding one JButton to link to the next interface.

Set the title of the JFrame as TIMETABLE, then set the default position when it is opened. Then add content of each JLabel.

JPanel1 add: l1 = new JLabel("Welcome to Timetable System!"); JPanel2 add: l2 = new JLabel("Please input the number of classes of each grade:"); tf = new JTextField(2); JPanel3 add: n = new JButton("next");

Then add a click-on event on the button, so that it can be clicked to jump to the second interface. The execution of the main function is also written in this class.

This interface is used to give users a brief introduction and allow users to set each grade's number of classes.

2 Set modules interface

The main function of the second interface is to set up each grade's weekly curriculum, in which the total number of courses per week must be 15.

This interface is written in class ModuleJFrame, set up a JLabel for each grade. Using a JCheckBox to display a module of each grade, and add a different JTextField after each JCheckBox. In this way, users can set up whether the class will take a course and the amount of each course.

In the interface, a default value is set for each module, it makes it more convenient for testing and users to really use them without having to manually set all values at one

time.

Add a JButton at the bottom of the interface, and add click events, click and the course scheduling will begin, and if success to get a result, jump to the third interface. If the total number of courses is incorrect or there still exist a hard conflict in the result, a different warning will be displayed to remind the user to reset it.

3 Timetable interface

This interface is the last interface to display the scheduled timetable, written in class TtJFrame. In this JFrame, the CardLayout is used. Add 6 timetables of each grade to the 6 JPanels, and add 6 JPanels to the interface by CardLayout. Click buttons which correspond to each grades timetable, the timetable will be showed separately.

The difficulty is that the resulting curriculum is shown by the course of each class from morning till night, involving a great deal of logic. First of all, the courses need to be put in 6 sets according to different grades.

I also used the JTable to display the timetables in each JPanel. Secondly, the courses are separated according to different classes, and each class's courses are inserted into JTable according to the time ID. Finally, the curriculum is displayed reasonably.

5.3 Evaluation

After the initial population is established, the fitness of these individuals needs to be evaluated and the fitness value is given. As we have learned above, the principle of optimizing the timetable is not to break the restrictions as much as possible. In violation of more constraints, the lower the fitness of individuals.

In the method of calculating the number of conflicts, the total number of conflicts increases by 1000 per violation of a hard constraint. For each violation of a soft constraint, 1-3 is added to the total number of conflicts which is based on the priority level, and finally the total number of conflicts is returned. Then add the method of

calculating fitness in the Ga class:

```
double fitness = 1000 / (double) (conflicts + 1000);
```

When the number of conflicts is 0, fitness is 1. If there is a hard conflict, then fitness¡0.5, otherwise there is only exist soft conflicts. Then add the evalPopulation method in the Ga class and calculate the fitness value for each individual.

There are two kinds of ending conditions: 1) generation number, 2) fitness value reach 1. Two conditions are both used while executing the loop.

5.3.1 Testing

Set the maximum number of generation to 2000 in the code, and then start the program. After you click the generate timetable button,

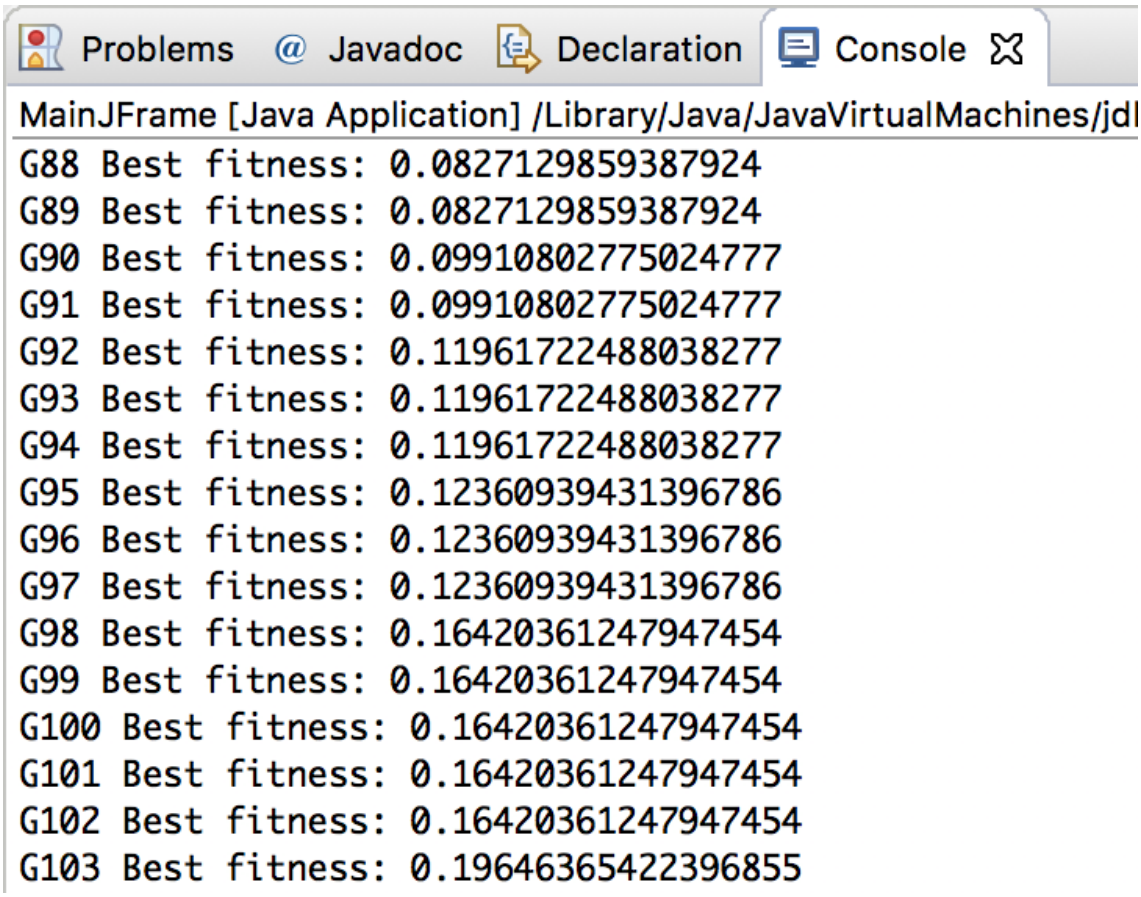

Figure 5.1: fitness increasing

the running process is shown in figure 5.1.

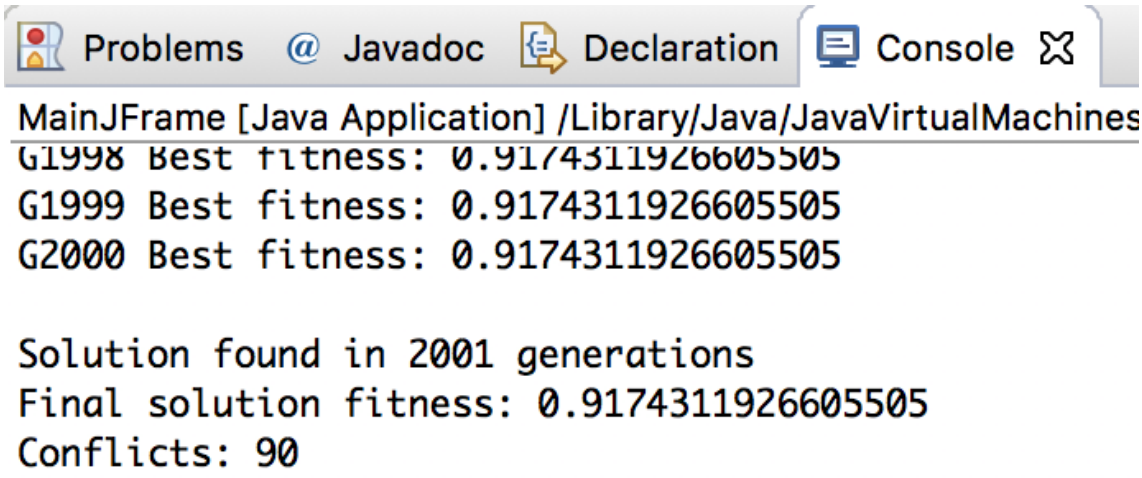

Figure 5.2: final result obtained

It can be seen that with the increase of generations number, fitness value continues to grow until the 2001 generation, as shown in figure 5.2.

As you can see, because there are only 90 conflicts in the final result, there is no hard conflict exists, so we have an available result. As the number of generation inceases, the number of soft conflicts continues to decrease, but it takes cost a lot of time.

As a result, this schedule system successfully gets the best available timetable with no hard conflicts.

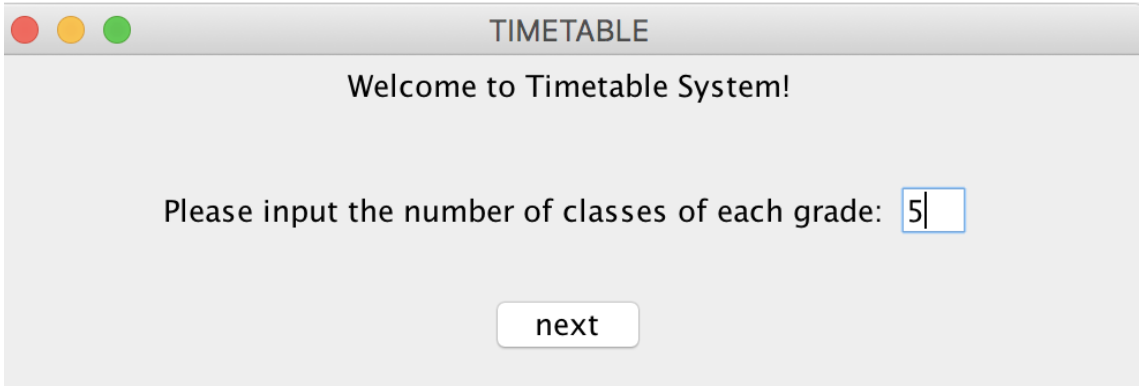

Figure 5.3: Set more classes of each grade

If many classes are set in each grade, such as Figure 5.3, it is difficult to get the results in the 2000 generation, so the timetable should not be displayed.

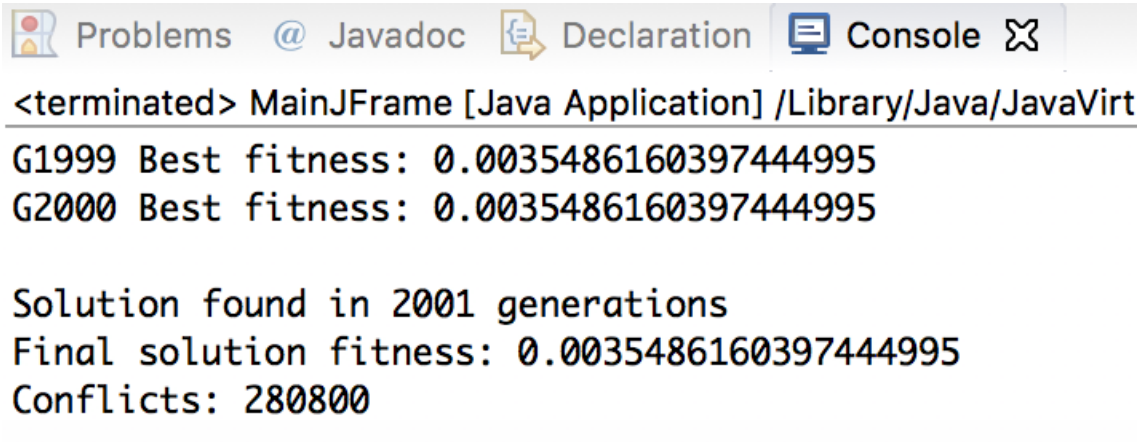

Figure 5.4: Result with 30 classes

As you can see in Figure 5.4, the fitness value is still very low and there are many conflicts during the 2000th generation, so the results are not available.

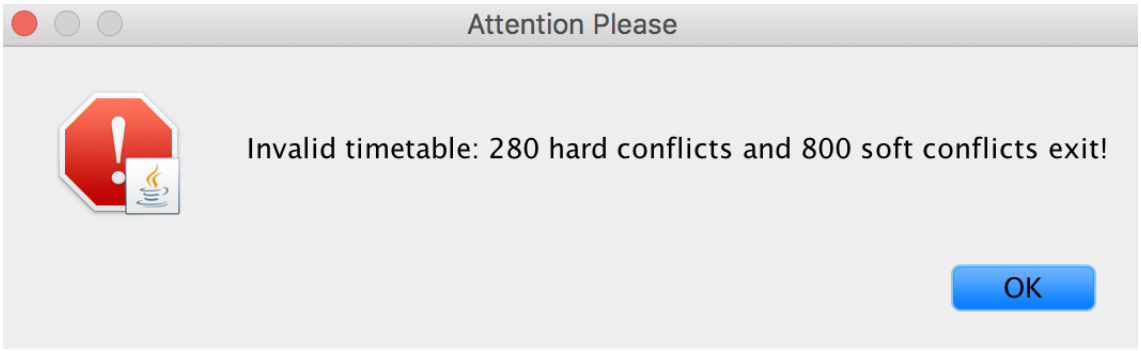

Figure 5.5: Hard conflicts exist

A warning will also be popped up, as shown in Figure 5.5, and the final results will not be produced. Since there is no available timetable obtained.

5.3.2 Self-evaluation

It took me almost two months to finish the project. In July, I read a lot of related literature to learn the options could be used, and then started to write the system in August. I first tried to write in language objective C, but I encountered a lot of problems that cannot be resolved, such as the return value type cannot be an array, etc., and eventually I gave up and wrote by Java instead. The structure and logic of the whole system are relatively clear, and there is no big problem in the process of writing this system.

I learned the difficulty and the key in solving this problem is to solve the existence of hard constraints and soft constraints, I read a lot of literature, in which there are many kinds of methods of using genetic algorithm to solve the and computing the total conflicts. At first I added only hard conflict calculations to the system, so each generation's fitness was affected only by hard conflicts. Later, when adding soft conflict calculations, I also thought of a number of ways to separate the importance of hard conflicts from soft conflicts and sort out different soft constraints according to importance. Finally, I modified the fitness calculation to get the results I wanted.

There was a lot of logic involved in the final interface to display the timetables, and I spent one day tuning it up and clearly showing it successfully.

Chapter 6

Results and discussion

6.1 Results

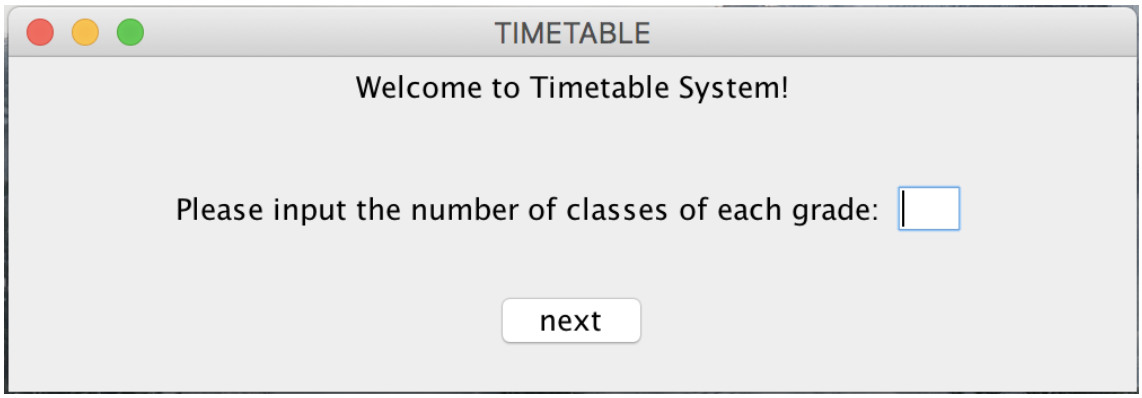

Figure 6.1: Welcome interface

Run the program, the first interface is shown in figure 6.1.

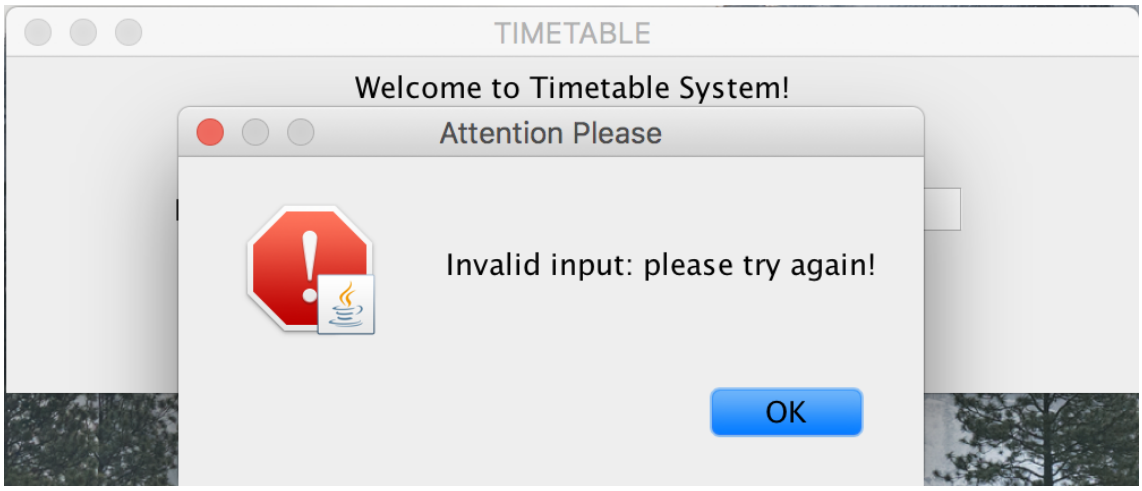

Figure 6.2: Attention1

If the class number is not set, a warning is issued: Invalid input, as shown in figure 6.2.

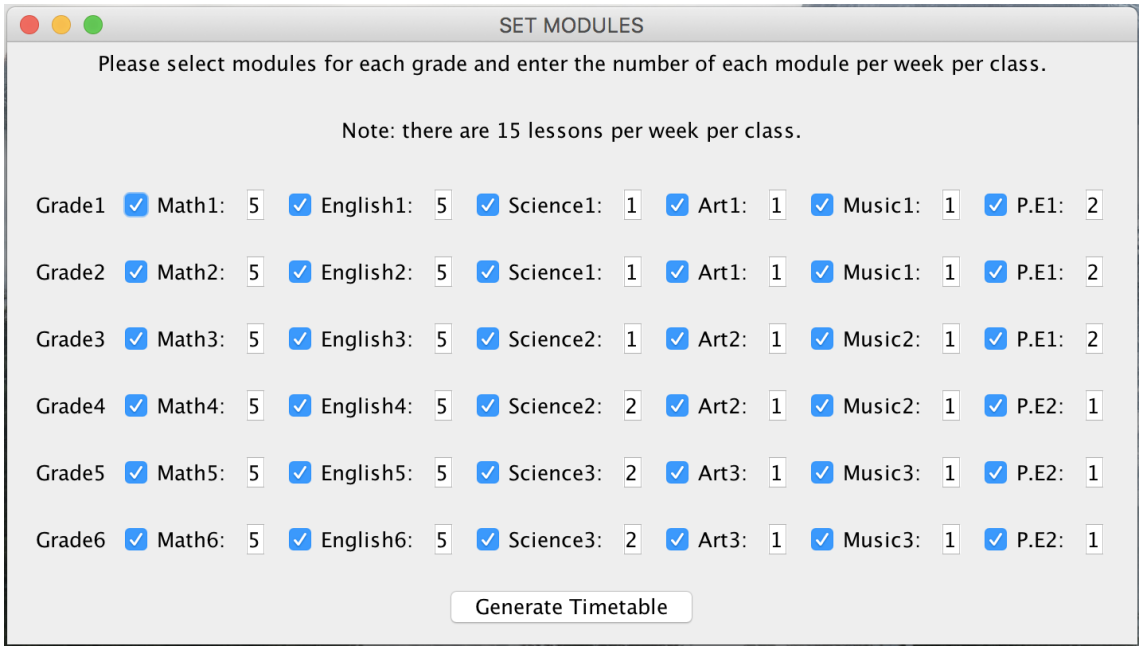

Figure 6.3: Set module distribution

Then we set the class number to 1, click Next button, and jump to the following interface: Fig 6.3.

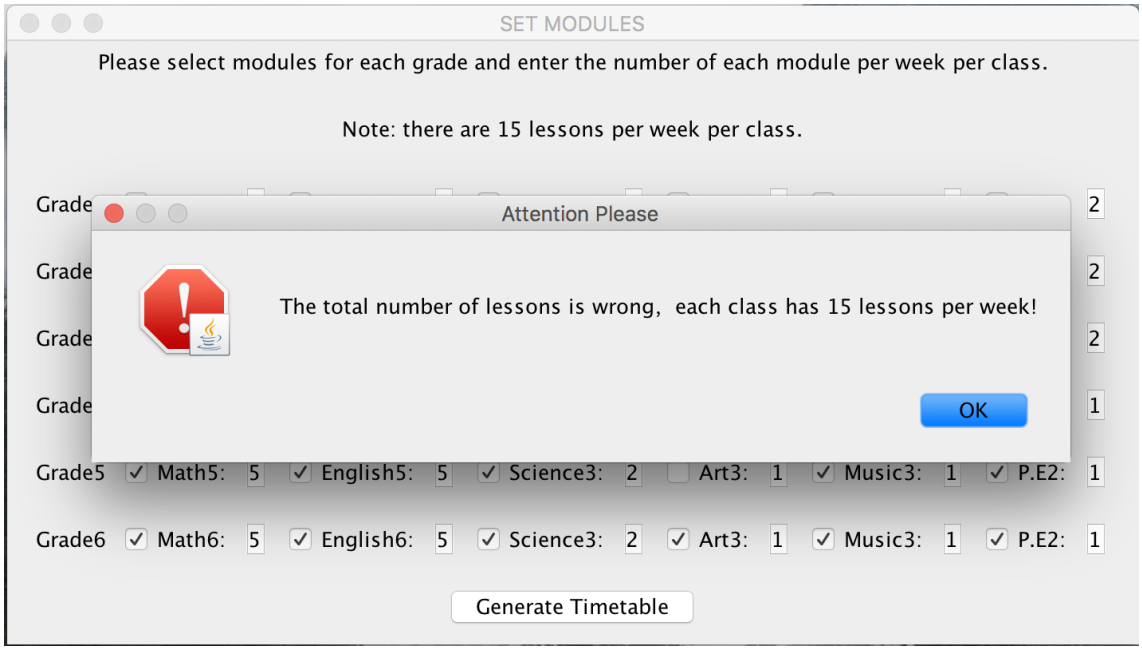

Figure 6.4: Attention2

In the interface is the default value we've written, you can make any changes, and then I cancel a JCheckBox check and press button to pop up the warning, as shown in figure 6.4.

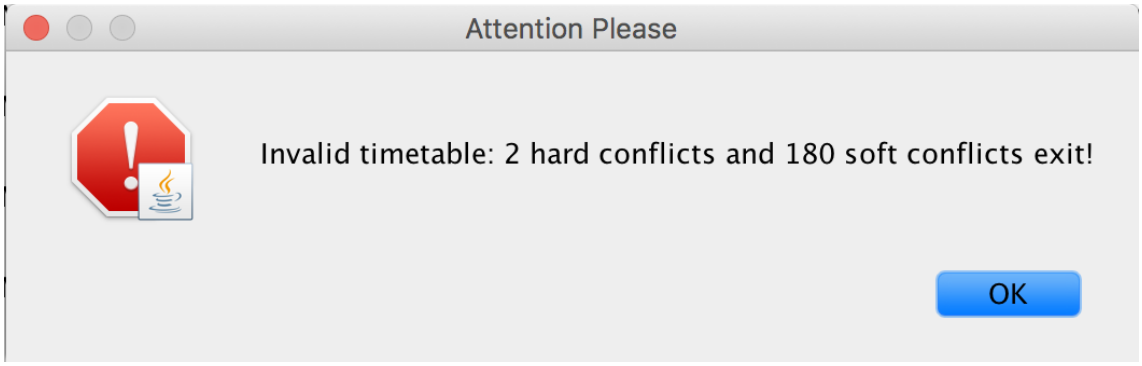

Figure 6.5: Attention3

Next, we set the class number to 2 and calculate the result, a warning shown in

figure 6.5:

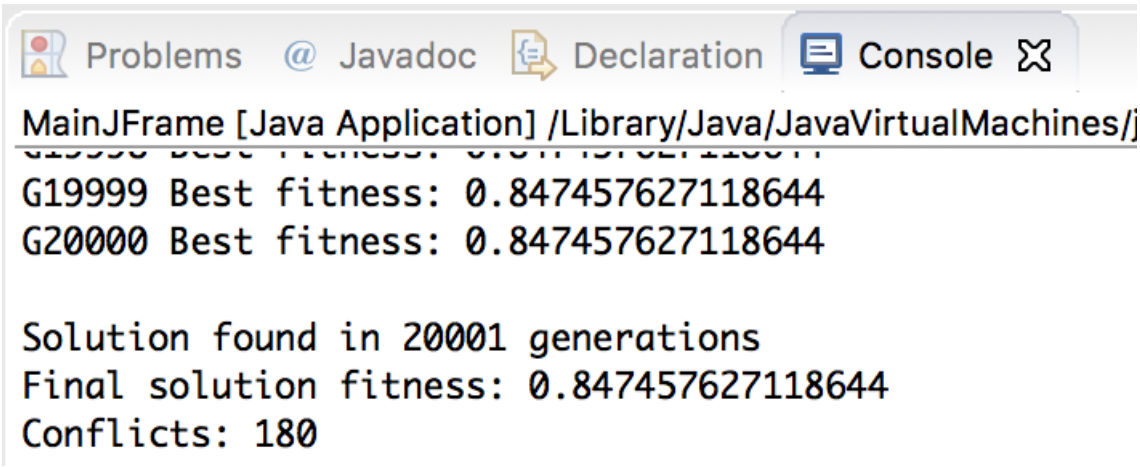

Figure 6.6: Results with 30 classes

It can be seen, because the condition of generation number is too small, we can't find the available results. Then, let's add generation number to 20000 and try again. The result is shown in Figure 6.6.

And the timetable shows as fig 6.7- fig6.12.

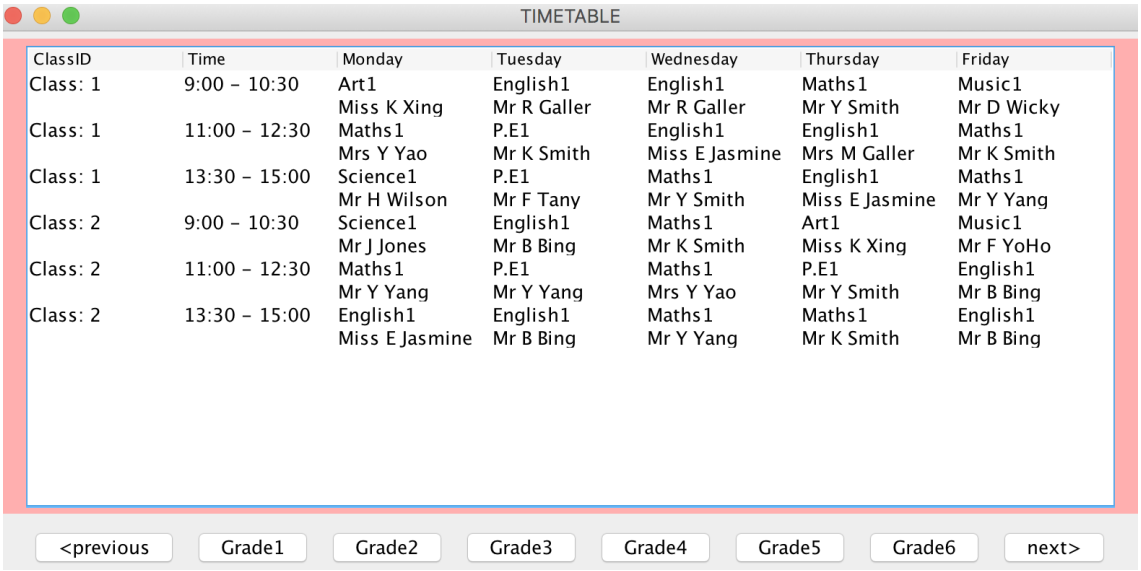

Figure 6.7: Grade1

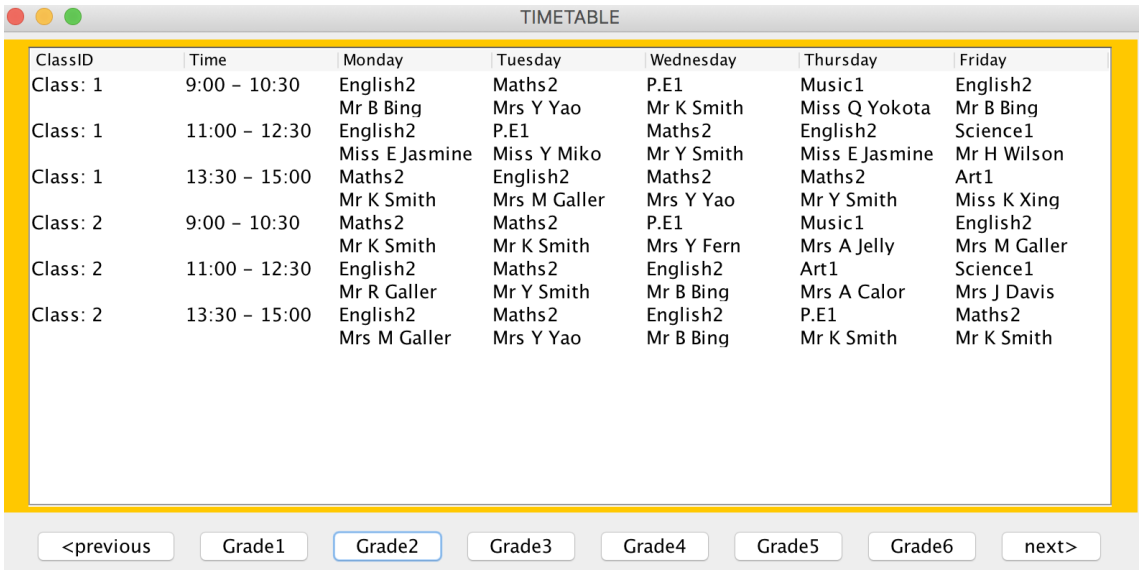

Figure 6.8: Grade2

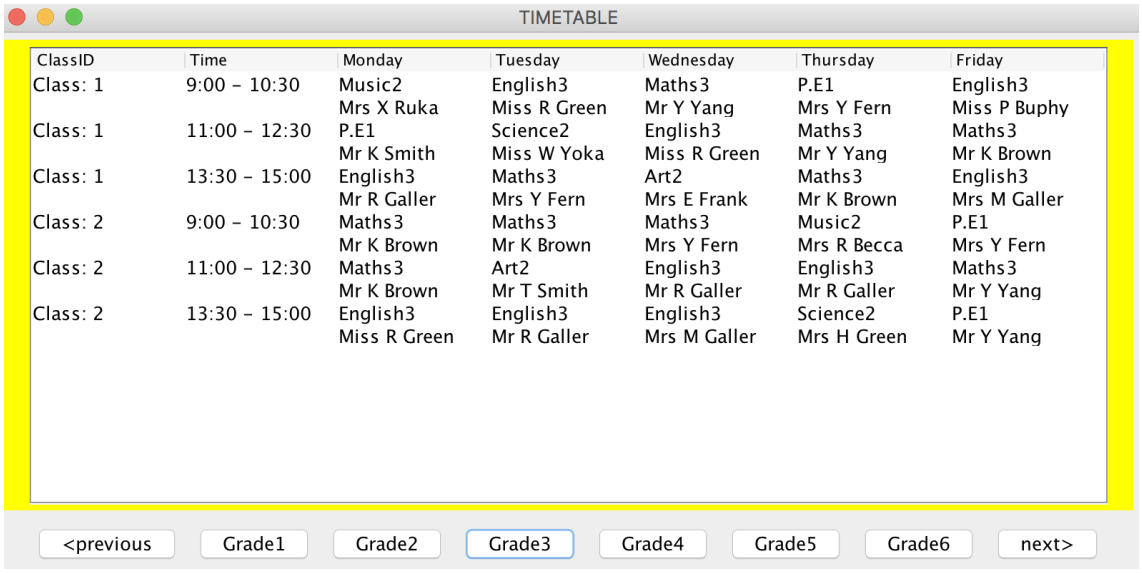

Figure 6.9: Grade3

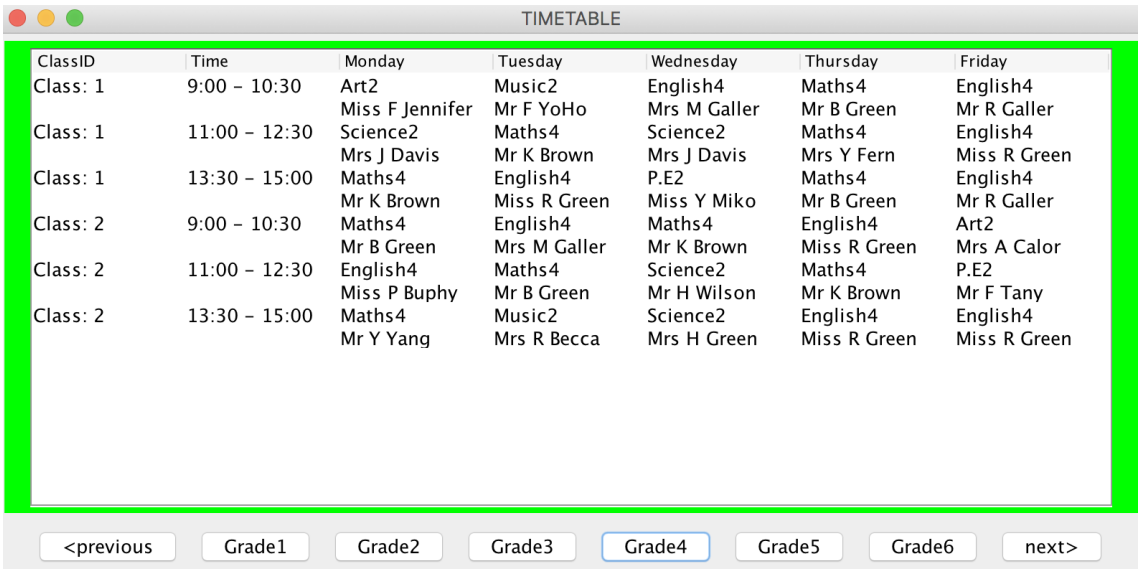

Figure 6.10: Grade4

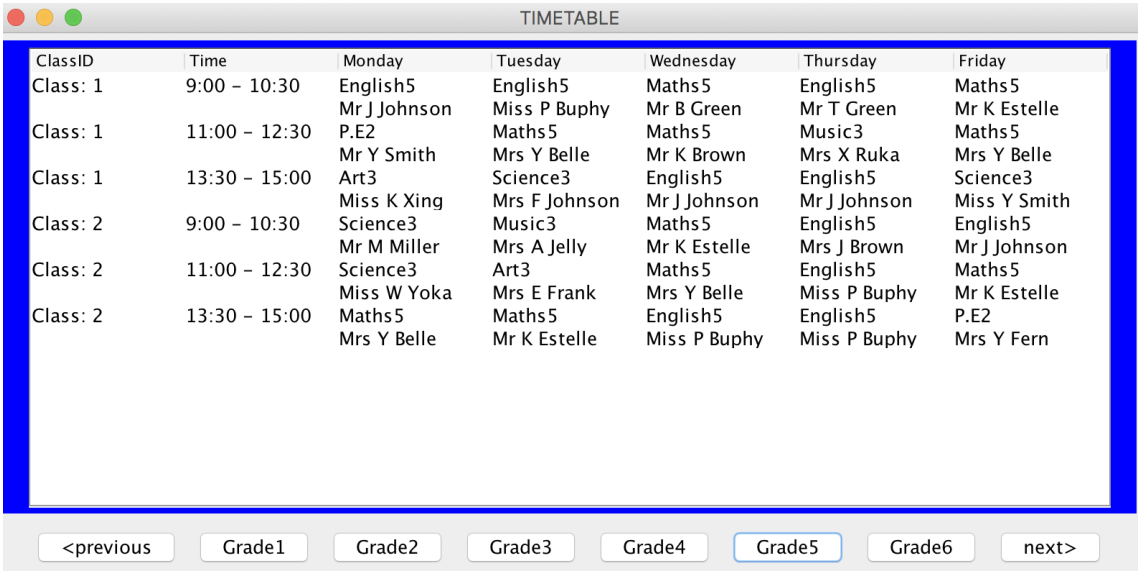

Figure 6.11: Grade5

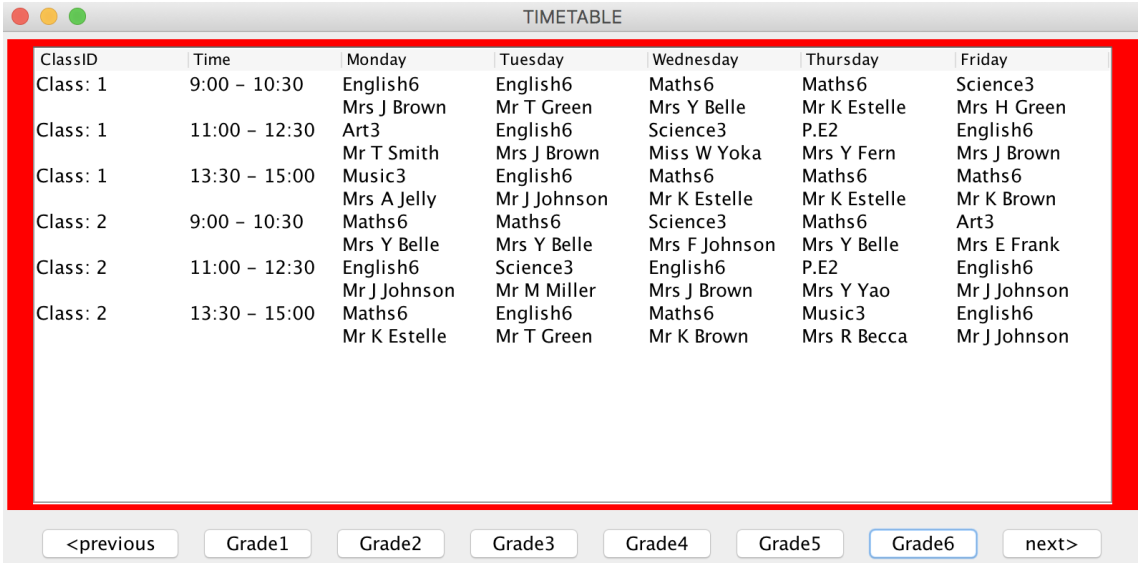

Figure 6.12: Grade6

6.2 Discussion

From the results, we can see that in the running process of the system, the situation that may arise is basically considered: the input box is not filled, hard conflict exists without getting reasonable results warning, course setting error pop-up warning, and the final result is separated by grades and classes.

And we can see from the timetable that we set two soft constraints: schedule English and Math class on the morning and try not to schedule Miss Yaos class on Monday morning and Friday afternoon. Thus, the hard constraints are all achieved and the soft constraints are implemented as much as possible, so the system is a system with a good quality timetable obtained. It realizes the function which we want to realize, and has certain practical value.

Chapter 7

Conclusion

7.1 Summary of content

During the completion of the project, firstly I learnt what the timetable problem is and the difficulties exist in this problem which is mainly focused on hard constrains and soft constraints. Then I read a lot of related literature, studied the algorithms that could be applied to solve the timetable problem, and finally the genetic algorithm is chosen to be applied in my system.

In applying genetic algorithms, I coded a single gene by all lessons for each week of one class. Each course consists of one time id and one teacher's ID, and uses a week's lessons of all classes to represent a chromosome. Then select chromosomes by tournament select method for crossover and mutation. In order to get better result, I used a function to calculate the fitness of each generation and get the last one with the best fitness, in which are no hard constraints and schedule the timetable by the priority of soft constraints.

After obtaining the result, I wrote a GUI interface to allow users to set the number of classes and the weekly module assignments per grade. If no feasible results are obtained, the number of hard conflicts existing will be displayed. And the final results will be presented through each class's timetable.

7.2 Summary of evaluation

In completing the project, I learned the difficulty of solving the timetable problem, and solve all the hard constraints and several soft constraints in my system. And I get the available timetable by applying genetic algorithm successfully.

I think the function to calculate the fitness of each generation is good. This method ensures that there is no hard conflict exist in the final result, and also schedule the timetable according to the priority degree of the soft constraints, this part is relatively successful. However, because there are fewer data in the system, it is difficult to get the result without hard conflict when the number of class increasing, but since it is a small primary school, we ignore this problem.

Finally, the timetable interface will show each timetable of different classes in different JPanel, the interface design is clear and simple, users can easily set up classes number and courses.

7.3 Outlook

If I had to do this project again, I will study the difficult point first as well, and read relevant literature to find available ideas and apply genetic algorithms to solve this problem. However, when I input data, I will write an interface in the program to import the use of the database, making the system easier to be used to add, delete and modify data.

The first one is adding the use of database that is mentioned above, which could make the system suitable for some bigger schools. The second one is to modify the interface, to make users can add, delete or modify the soft constraints by themselves, which would make the system more interactive and realistic.

Bibliography

[1] Burke, E. K., Newall, J. P., and Weare, R. F. A memetic algorithm for university exam timetabling. 241–250.

[2] C, X., M, X., L, J., and S, J. Algorithm research of university timetabling system under multiple constraints. *Computer Knowledge and Technology 3*, 27 (2008), 1958–l959.

[3] Cooper, T. B., and Kingston, J. H. The complexity of timetable construction problems. 281–295.

[4] Dorigo, M., Maniezzo, V., and Colorni, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26*, 1 (1996), 29–41.

[5] Even, S., Itai, A., and Shamir, A. On the complexity of time table and multi-commodity flow problems. 184–193.

[6] Garey, M. R., and Johnson, D. S. Computers and intractability.

[7] Glover, F. Tabu searchpart i. *ORSA Journal on computing 1*, 3 (1989), 190–206.

[8] Glover, F. Tabu searchpart ii. *ORSA Journal on computing 2*, 1 (1990), 4–32.

[9] Gotlieb, C. The construction of class-teacher timetables. 73–77.

[10] Gutjahr, W. J. A graph-based ant system and its convergence. *Future generation computer systems 16*, 8 (2000), 873–888.

[11] Hertz, A. Tabu search for large scale timetabling problems. *European journal of operational research 54*, 1 (1991), 39–47.

[12] Jacobson, L., and Kanber, B. Genetic algorithms in java basics.

[13] L, M. Memetic algorithm research progress. *Automation technology and Application 26*, 11 (2007), 1–4.

[14] Lawrie, N. L. An integer linear programming model of a school timetabling problem. *The Computer Journal 12*, 4 (1969), 307–316.

[15] Moscato, P. Genetic algorithms and martial arts towards memetic algorithm. *Calteth Concurrent Computation Program Report 826* (2004).

[16] Sim, K. M., and Sun, W. H. Multiple ant-colony optimization for network routing. 277–281.

[17] X, K., and Z, S. Design of an automatic course scheduling system based on genetic algorithm. *Computer Security*, 10 (2007), 9–12.

[18] Xu, P. Application of simulated annealing algorithm in automatic course scheduling system. *Journal of Jinggangshan University: Comprehensive Edition 29*, 6 (2008), 35–37.

[19] Z, L. Research and design of courses arrangement based on ant colony algorithm.

[20] Z, Y. Design and implementation of timetabling system based on java.

Appendices

Appendix A

An Appendix of Some Kind

Appendix B

Another Appendix