Criterion B—Design

Structure Chart

The structure chart shows three main functions of the system that are processed in sequence and their sub-functions. The first function is ***inputData***, which asks a user to input threes files that contain data of students, teachers, and courses. The second function is ***scheduling***, which processes data from imported files and generates six groups that satisfy the success criteria with an optimization algorithm (Genetic Algorithm). The third function is ***outputData***.

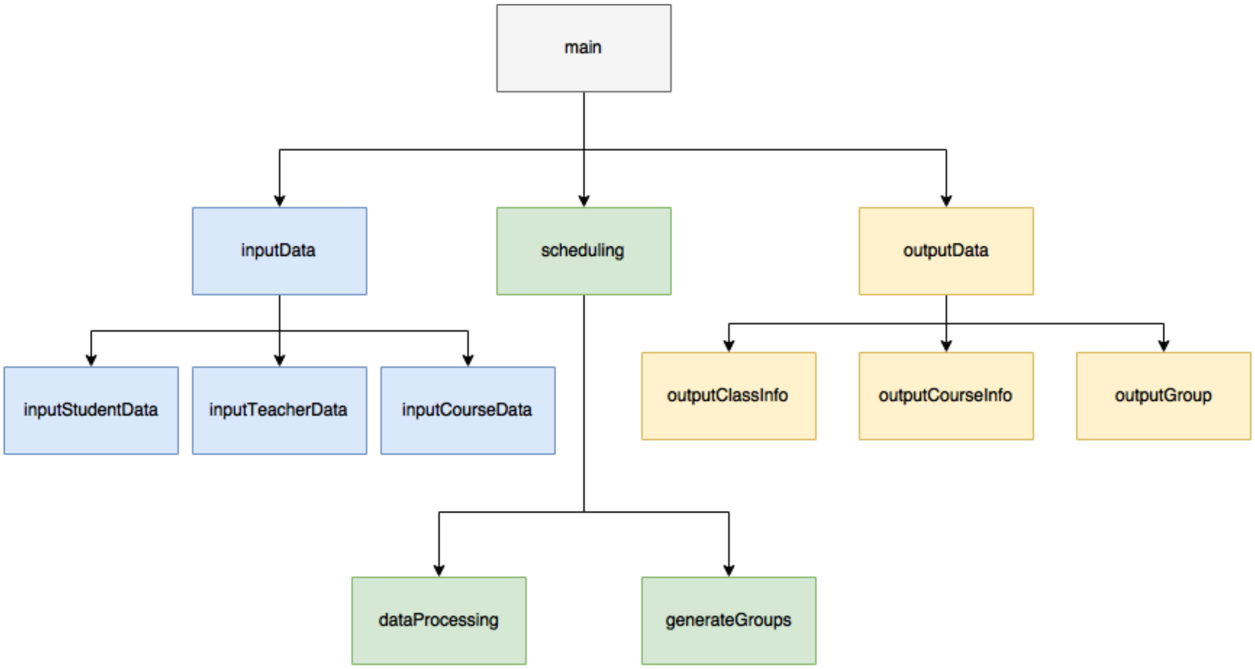


Figure B.1 Structure chart

Data Structure Design

I design three Excel files formatted as following in order to input all necessary information:

Teacher Information	
Name	Subject
…	…

Course Information		
Course	Level	Maximum Class Number
…	…	…

Student Information											
Student Name	Subject 1	Level 1	Subject 2	Level 2	Subject 3	Level 3	Subject 4	Level 4	Subject 5	Level 5	Subject 6
...	...	...	...	...	...	...	...	...	...	...	...

I group the data into fields of three corresponding classes—***Student***, ***Teacher***, and ***Course***. I also create a ***IBClass*** class. I use Java collections to hold sets of ***Student***, ***Teacher***, ***Course*** objects.

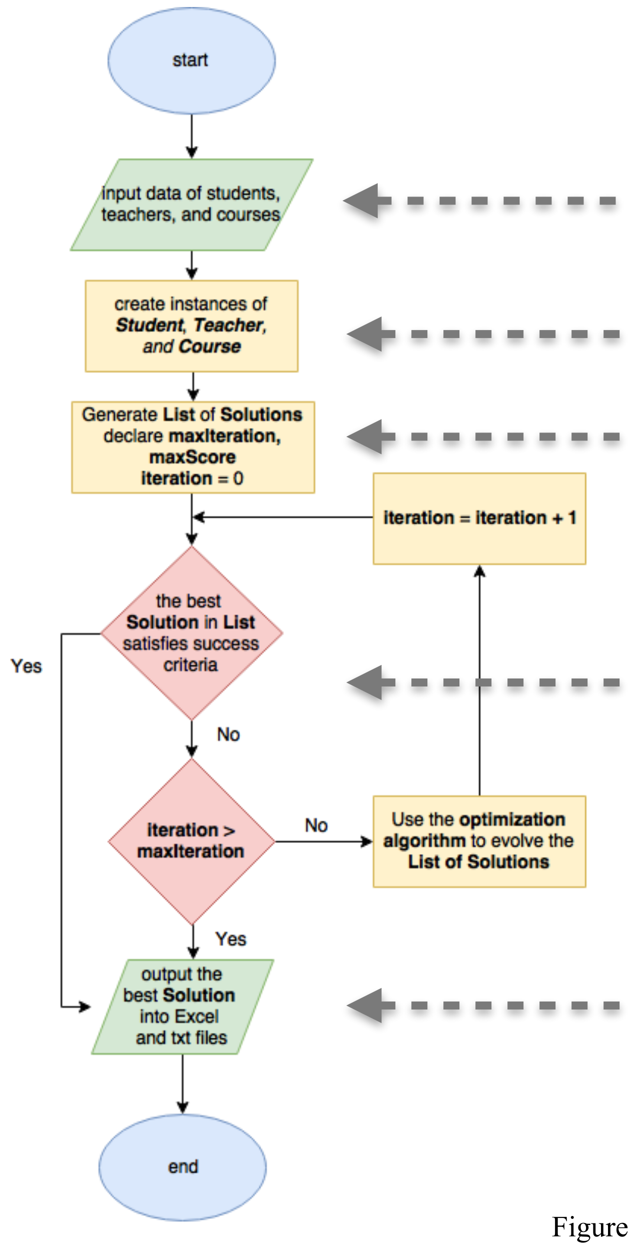
Student	Teacher	Course	IBClass
-name:String -courseList: Collection	-name:String -subject:String	-subject:String; -level:String; -maxClassNumber:int; -classNumber:int; -classList:Collection; -studentList:Collection; -teacherList:Collection;	-subject:String -level:String -group:int -id:int -studentList: Collection

Overall Flowchart and Pseudo-code

After inputting data and instantiating objects, the system creates an initial set of solutions. Then the system uses an optimization algorithm, genetic algorithm, to better the solutions with ***evolve()*** method. If the best solution in the solution sets is perfect or the system goes through sufficient iterations, results are outputted.

Overall Flowchart

Overall Pseudo-code


start
input data of students, teachers, and courses
create instances of Student, Teacher, and Course
Generate List of Solutions declare maxIteration, maxScore iteration = 0
iteration = iteration + 1
the best Solution in List satisfies success criteria
Yes
No
iteration > maxIteration
No
Use the optimization algorithm to evolve the List of Solutions
Yes
output the best Solution into Excel and txt files
end

```
input studentData, teacherData,
and courseData

instantiate Student, Teacher,
and Course objects

Generate Collection of Solution
set iteration to 0
declare maxIteration, maxScore

while List.getBestSolution().getScore() <maxScore AND iteration < maxIteration
      Population.evolve()
end loop

output Solution
```

Figure B.2

Flowchart and Pseudo-code of the Optimization Algorithm (Genetic Algorithm)

I plan to use genetic algorithm, an optimization algorithm, to generate solutions. In genetic algorithms, a collection of ***Solution*** objects are created, and the collection is a ***Population***. The number of ***Solution*** objects in ***Population***, **populationNumber**, is pre-defined. The algorithm assesses and gives a score to each ***Solution*** object. The ***Solution*** with highest score will be kept, and others ***evolve***. Iterations continue until the system gets a perfect solution whose score equals to the pre-defined ***maxScore*** or iteration number exceeds the pre-defined maximum.

Pseudo-code

```
set count to 0
Population = Solution [populationNumber]
while count < populationNumber
      Population[count] = new Solution()
      count = count+1
end loop

highestScore = 0, bestSolution = null
count=0, iteration = 0
declare maxIteration, maxScore
while highestScore<maxScore AND iteration < maxIteration
      while count < populationNumber
            score = Population[count].getScore()
            if (score > highestScore)
                  highestScore = score
                  fittest = Population[count]
            end if
            count = count+1
      end loop

      Population[0] = fittest
      set count to 1
      while count < populationNumber
            Population[count]=Population[count].evolve()
            count = count+1
      end loop
end loop
```

Flowchart

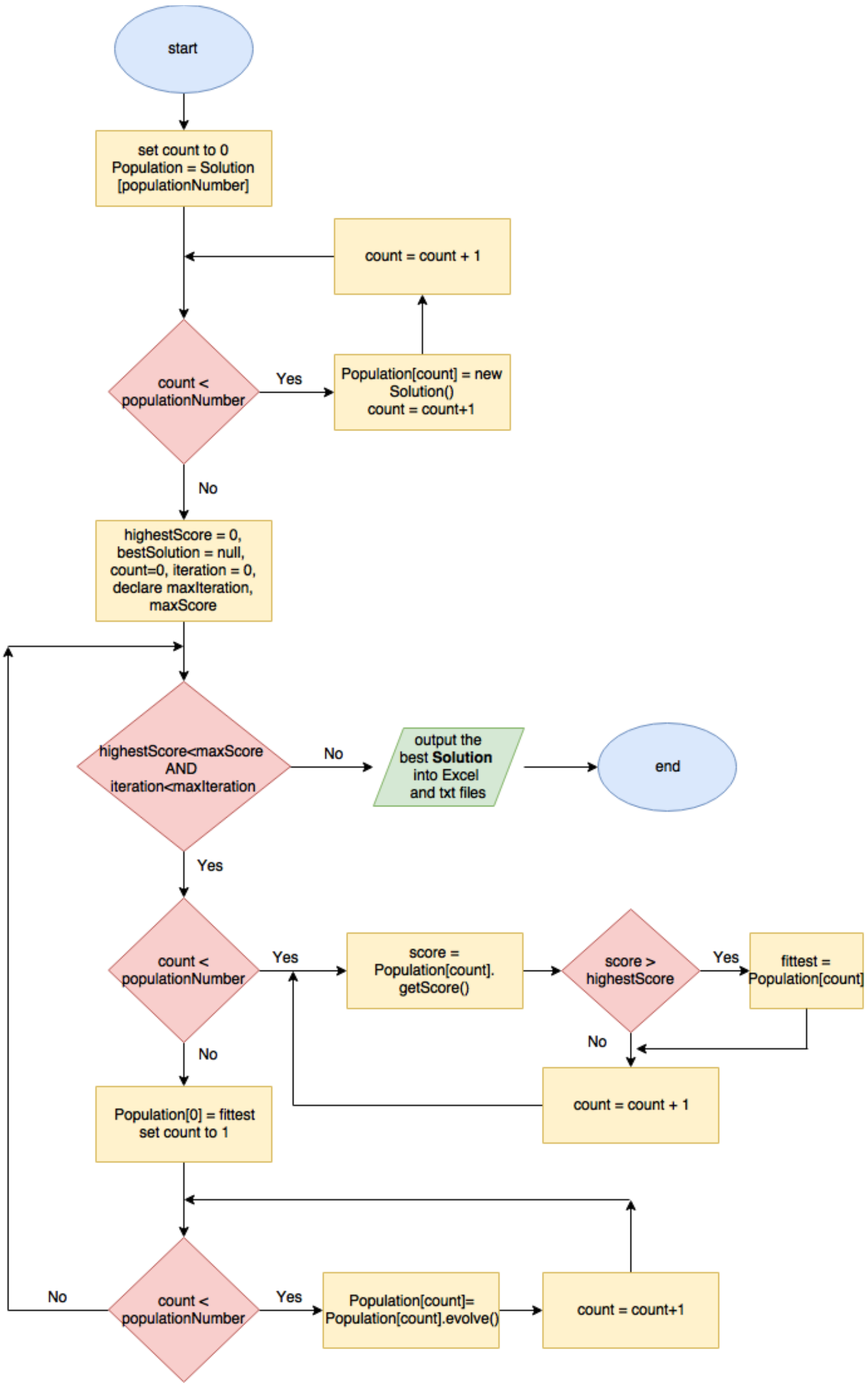
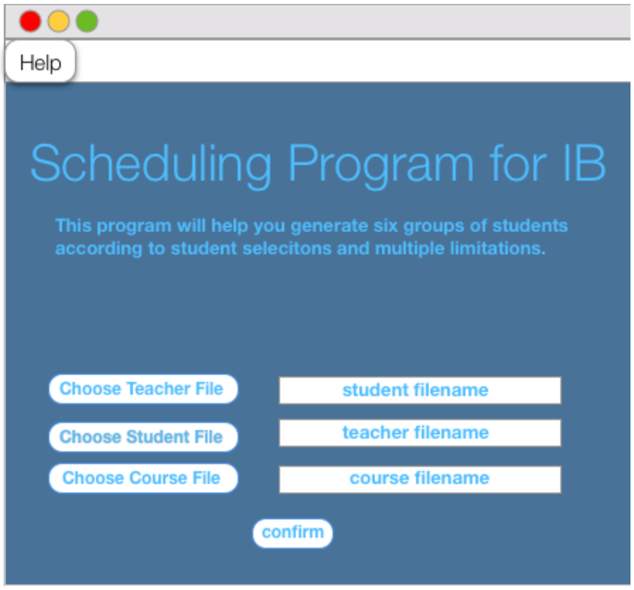


Figure B.3 Flowchart of the Genetic Algorithm

User Interface Design

The first page is a welcome page and asks a user to select three files to input data. Also, there is a "Help" button on the top-left corner for instructions.



Scheduling Program for IB

This program will help you generate six groups of students according to student selecitons and multiple limitations.

Figure B.4 GUI page 1

The second page is a progressing page. There is a progression bar and a percentage showing the perfectness of the current result. A "Stop" button enables a user to stop the program immediately and output the best available result (which may not be the perfect result). A label informs the user whether the process is completed or uncompleted.

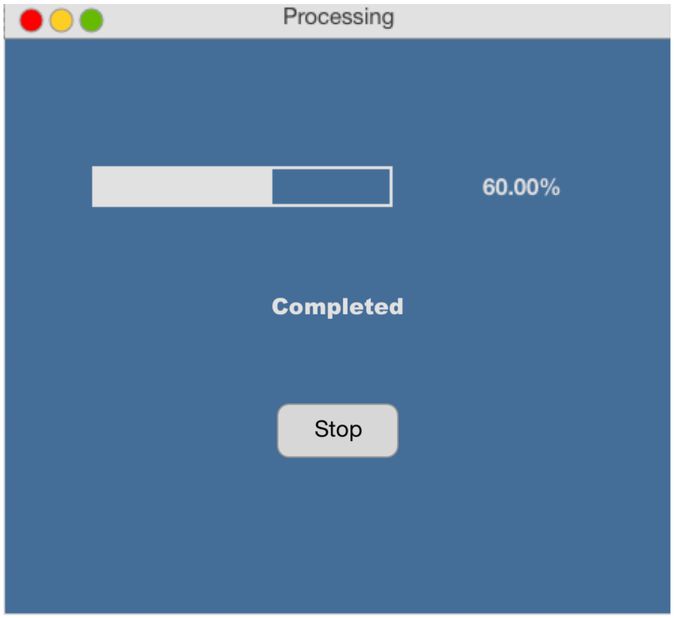


Figure B.5 GUI page 2

Test Plan

Test No.	Description	Input	Expected Outcome	Actual Outcome	Figure No.
Unit Testing					
1	Instantiate ***Student***, ***Teacher***, ***Course***, and ***IBClass*** objects	Parameters of correct data types	Objects are instantiated successfully	Objects are instantiated successfully	N/A
2	***toString()*** method of ***IBClass*** class	Instantiated ***IBClass*** objects	Clearly output the name, size, group, consisting students of each ***IBClass*** object	Certain words attach to each other; the layout is not perfectly clear	1
3	***toString()*** method of ***Course*** class	Instantiated ***Course*** objects	Clearly output the name and corresponding classes of each ***Course*** objects	Certain words attach to each other; the layout is not perfectly clear	2
Data Testing					
4	Normal data	Formatted Excel files with correct information of student selections, teacher availability, and course information	The system is able to process data and instantiate respective objects according to the data	Meet some run-time errors that are fixed	N/A
5	Abnormal data	Incorrectly formatted information (a String is written instead of a number)	The system outputs an error message, "please fill in all the information and correct the format in XXX file" (XXX is the corresponding file)	Error message displayed	3
6	Extreme data	The number of students/ teachers/courses claimed is larger than the actual number	The system outputs an error message, "Please check if the number of XXX is correct" (XXX is the corresponding information)	Error message displayed	4
Functional Testing					
7	Run-time errors of the genetic algorithm	Instantiated Student, Teacher, Course objects	The algorithm is able to run without reporting errors	Meet some run-time errors that are fixed	N/A
8	Logic errors of the genetic algorithm	Instantiated Student, Teacher, Course objects	The algorithm is able to gradually optimize the solution	The algorithm successfully optimizes the solution gradually	N/A
Integration Testing					
9	Input files through the main GUI window	Three Excel files containing information of students, teachers, and courses	The user can easily identify the buttons to import, confirm, and find help; algorithm starts after the user confirms; an error message is displayed if a file is missing	The layout is clear; algorithm starts after each file is correctly inputted; error message is displayed when a required file is not inputted	5
10	Help menu in the main GUI window	Click on the button “Help”	A clear instruction dialogue appears	A clear instruction dialogue appears	6

11	The GUI window during Processing and its help menu	Three Excel files containing information of students, teachers, and courses	The window shows a progression bar indicating the level of competence	The window displays the progression bar	7
12	“Stop” function of the program	Click on the button “Stop”	The algorithm will stop and output the available best results	The algorithm stops and outputs result	N/A
13	The outputs	Three Excel files containing information of students, teachers, and courses	Results are clearly formatted in text and Excel files	Results can be found in the designated directory and the results are well formatted	8
Success Criteria Testing					
14	Student conflicts	Three Excel files containing information of students, teachers, and courses	Each student has only one class in each group	The algorithm almost succeeds to avoid student conflicts	N/A
15	Class size	Three Excel files containing information of students, teachers, and courses	The size of each class is more than 3 and less than 20, unless the total number o students choosing the course is less than 3	The algorithm almost succeeds to maintain appropriate class sizes	N/A
16	Teacher conflicts	Three Excel files containing information of students, teachers, and courses	If a subject has only one teacher, there cannot be two classes of the subject in the same group;	The algorithm succeeds to avoid teacher conflicts for the most time	N/A
17	Marks for imperfect outputs	Three Excel files containing information of students, teachers, and courses	Imperfect result is marked out; the meaning of different types of marks is explained	Different marks are provided for conflicts; a legend that explains the meaning of marks is provided	9
Beta Testing					
18	Provide the program to the client	Three Excel files containing information of students, teachers, and courses of Grade 12 Year 2016-2017	A user will test the system with data from last year and evaluate the effectiveness of the system	User asks for more instructions in the system	Appx. B
Acceptance Testing					
19	Offer the finalized program to the client for feedback	Three Excel files containing information of students, teachers, and courses of Grade 11 Year 2018-2019 (next year)	I will help the user to use the system in the next year when he needs to generate schedules	Feedback received: “The current result produced by your system was pretty useful. The system is able to generate a nearly perfect solution within half an hour. Based on that and the marks indicating existing conflicts, I can work on the solution given to get a workable solution rather quickly.”	Appx. B