**CHAPTER THREE**

**METHODOLOGY**

**3.0 METHODOLOGY**

Making a time table is one of those Numerical Problems (NP) hard problems. The problems can be solved using a heuristic search algorithm to find the optimal solution, but it only works for simple cases. For more complex inputs and requirements, findings a considerably good solution can take a while, or it may be impossible. This is where cultural algorithms come into the game which comprises of **Belief space** and the **Population space**. To tackle the population aspect, Genetic algorithm is used. The belief space is the place where the information on the solution of the problem is refined and stored. The information stored is updates from time to time as new generation is created and used to generate another time table which will be better than the previous time table.

**3.1 CULTURAL BASED GENETIC ALGORITHM**

Cultural Based Genetic Algorithm is a combination of version space algorithm (a binary based genetic algorithm) and Knowledge Base which is where information acquired by generations is stored, and which is accessible to the current generation.

**3.2 IMPLEMENTATION**

The below contains the step-by-step procedure taken to generate the Nurse roster:

1. Data requirements.
2. From the requirements, the hard and soft constraints are determined.
3. Determining the fitness function using the hard and soft constraints.
4. Deduction of a flowchart to simplify problem.
5. Designing a suitable interface that will look like a time table.
6. C# programming language is used to implement the genetic-based cultural algorithm.
7. Debugging and testing of the program.

**3.3 OBJECTS OF A TIME TABLE**

Lecturer: the lecturer class has an ID and the name of the lecturer.

Students Group: The Students Group class has an ID and the name of the Student group.

Classroom: the Classroom class has an ID and the name of the classroom, as well as the number of seats and information about equipment (compuetrs). If the classroom has computers, it is expected that there is a computer for each seat. IDs are generated internally and automatically.

Course: the course class has an ID and the name of the course.

Class: CourseClass holds a reference to the course to which the class belongs, a reference to the lecturer who teaches, and a list student groups that attends the class. It also stores how many seats (sum of student groups sizes) are needed in the classroom, if the class requires computers in the classroom and the duration of the class (in hours).

**3.4 CASE STUDY**

In this project, Ladoke Akintola University of Technology (LAUTECH) will be used as the study and we will focus on the Department of Mechanical Engineering to maintain simplicity in the system. The algorithm can be further extended to the whole school. It has been noticed that there are usually problems with the timetable (TIMTEC no offence). Problems like clashes, capacity miscalculation, double fixing of leacture at same time so on are not uncommon.

**3.5 PROPOSED SYSTEM**

System to be developed will be implemented using C#, a .NET Language with some readymade Genetic and Cultural Based Algorithm alongside its API relating to Numerical Computation and Mathematical operations.

**3.6 CONSTRAINTS**

HARD

* No student can attend more than one lecture at the same time.
* No lecture can teach more than one lecture at the same time.
* No Room can occupy more than one lecture at the same time.
* No Room can be assigned a lecture with more students than its capacity
* .Some courses are scheduled in blocks of more than one hour, there restrictions must be respected

SOFT

* Minimize the use of early morning (8 A.M) and Late evening hours (18-20)
* Minimize the use of Friday 1P.M – 2P.M hours and 4PM – 7PM hours slots to allow for Muslim prayers and Adventists Sabbath day respectively.
* Minimize continuous lectures/blocks of the same course in a day.
* Student should not have only one lecture in any given day

**3.7 DATA USED FOR THE PROJECT**

**3.7.1 LECTURER DETAILS**

|  |  |
| --- | --- |
| S/N | Lecturer Name |
| 1 | Dr. Oyetunji |
| 2 | Dr. Anawumi |
| 3 | Prof. Lucas |
| 4 | Dr. Oladeji |
| 5 | Dr. Pius |
| 6 | Dr. Sangotayo |
| 7 | Dr. Durowoju |
| 8 | Dr. Ashafa |
| 9 | Dr. Aderibigba |
| 10 | Dr. Ajayeoba |
| 11 | Dr. Olaoye |
| 12 | Dr. Itabiyi |
| 13 | Dr. Mudashiru |

**3.7.2 COURSE DETAILS**

**300 LEVEL COURSE DETAIL**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | MEE 313 | MANUFACTURING TECHNOLOGY | 2 |
| 2 | MEE 313 | MANUFACTURING TECHNOLOGY | 2 |
| 3 | MEE 321 | STRENGTH OF MATERIAL 2 | 3 |
| 4 | MEE 323 | MACHINE DESIGN 1 | 3 |
| 5 | MEE 333 | MECHANICS OF MACHINES 2 | 2 |
| 6 | MEE 361 | APPLIED THERMODYNAMICS 1 | 2 |
| 7 | MEE 391 | METALLURGY | 2 |

**400 LEVEL COURSE DETAIL**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | MEE 423 | MACHINE DESIGN 2 | 3 |
| 2 | MEE 431 | MECHANICAL VIBRATIONS | 3 |
| 3 | MEE 441 | AUTOMATIC CONTROL SYSTEM | 2 |
| 4 | MEE 451 | APPLIED FLUID MECHANICS | 2 |
| 5 | MEE 461 | APPLIED THERMODYNAMICS 2 | 2 |
| 6 | MEE 471 | HEAT & MASS TRANSFER | 3 |
| 7 | MEE 481 | MECHANICS OF METAL FORMING | 2 |

**500 LEVEL COURSE DETAIL**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | MEE 541 | THERMAL ENGINES | 3 |
| 2 | MEE  551 | FLUID MACHINERY | 3 |
| 3 | MEE 573 | ADVANCED HEAT TRANSFER | 3 |
| 4 | MEE 583 | REFRIGERATION & AIR CONDITIONING | 2 |
| 5 | MEE 313 | MANUFACTURING TECHNOLOGY | 2 |

**3.8 PERFORMANCE METRICS FOR SOFTWARE**

The evaluations of the performances for the software generated are listed below:

* **Time complexity:** it is the average time the application takes to generate a solution. For each iteration, the time is noted down and the average time is calculated from these process times.
* **Computational complexity:** this involves halstead’s complexity and cyclomatic complexity. For halstead’s complexity, we find the program volume, program length, program difficulty, program vocabulary and effort while for cyclomatic complexity, it can be derived from the flowchart of the algorithm used, in some cases some compiler has these functions (for example, visual studio used for writing C-sharp programming language).
* **Maintainability index:** Calculated with certain formulae from lines-of-code measures, McCabe measure and Halstead measures. The higher the maintainability index the higher the rate at which the software can be maintained.

**3.9 PSEUDO-CODE OF A CULTURAL ALGORITHM**

1. Generate the initial population
2. Initialize the belief space
3. Evaluate the initial population
4. Repeat

* Update the belief space (with the individuals accepted)
* Apply the variation operators (under the influence of the belief space)
* Evaluate each child
* Perform selection

1. Until the end condition is satisfied

**3.10 FLOWCHART**

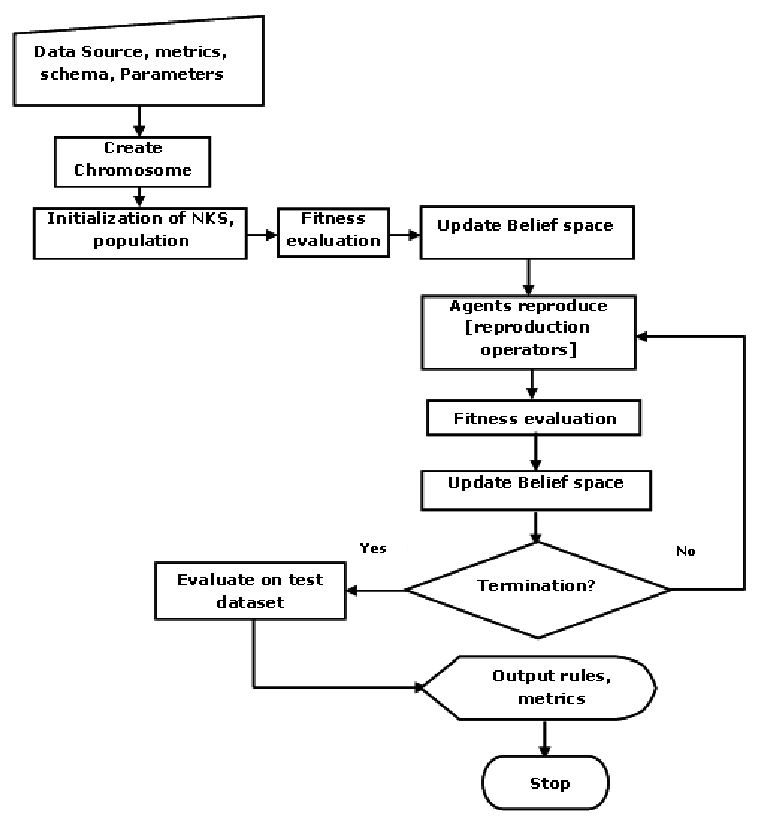
****

Figure 3.1 Flow chart for Genetic Algorithm