UNIVERSITY TIMETABLE SCHEDULING USING METAHEURISTIC ADAPTIVE-ELITIST GENETIC ALGORITHM

A CS Thesis Project

Presented to the Faculty of

AMA COMPUTER COLLEGE

LAS PIÑAS

In Partial Fulfillment of the

Requirements for the Degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

Researcher:

Celiz, Edcel A.

April 2018

DEDICATION

This research is dedicated to myself and fellow software developers. To remind that the question is not “can I make this?” but “how long till I make this?”

ACKNOWLEDGEMENT

I would like to acknowledge and thank the following:

Our God Almighty, for making all of this happen and serving as an inspiration towards the future.

My beloved parents, Maribel and Carlos Celiz, and brother, Christopher Celiz, who have always supported me in all their capabilities and served as my inspiration for my success in life.

My beloved partner, Araceli Liza, who have expressed her endless support, patience and love to me all throughout the research.

AMA Computer College Las Piñas' dean, Engr. Michelle Pullon, who made my research possible and guided my towards its development.

AMA Computer College Las Piñas' college faculty who have shown their support and knowledge which made the research possible.

Research panelists for creating right judgment and imparting their professional views on the research.

My friends and peers who have shown their support and help in times of distress.

TABLE OF CONTENTS

Figures

Tables

CHAPTER 1

THE PROBLEM AND ITS BACKGROUND

Introduction

Timetable scheduling is the process of creating timetables that fit the constraint of the scenario. It is used in a magnitude of industry from scheduling transportations up to creating complex schedule for highly optimized automated factories. Majority of small scale scheduling are done manually while larger operations require computer assisted scheduling.

Artificial intelligence is one of the rising computing solution due to an increase in computing power. It can be applied to different types of problems which can help optimize existing solutions or create never been tried solutions due to multiple limitations. Artificial intelligence helps create solutions for non-deterministic polynomial time problems which businesses and organizations will always have.

Genetic algorithm is a metaheuristic that mimics the process of natural selection. It can be performed in multiple different ways with different types but it will all follow the same concept. This research aims to create an artificial intelligence through the use of evolutionary algorithm, specifically genetic algorithm combined with adaptive and elitist traits that can generate a university schedule timetable with the goal of

generating a valid and as optimal as possible solution with certain constraints.

Background of the Study

Scheduling classes in an institution is one of the tasks that can be categorized under operations management. Operations management aims to maximize efficiency in a certain area. Scheduling classes is often done by humans which proves to be efficient but does not mean it is perfect. Using computers to find the best solution for this problem using conventional method is extremely inefficient. Computers were therefore ignored for a time when solving such problem. However, rise in computing power and usage has opened ways in solving scheduling problems.

Operations management is often considered as backbone of many companies. Getting the most efficient work means higher profit. In an educational institution scenario, getting the most efficient schedule does not only help reduce expense but mainly to cater to students. Often, better schedules give students and instructors better control on their time. Creating schedules is done better by humans than conventional computing techniques because humans have powerful brain to assess constraints and combinations better. However, scheduling is known to be non-deterministic polynomial time (NP) complete problem where in order to find the best solution, every possible combinations should be executed. Humans cannot compute every possible combination and therefore solves

the problem by just filling up and making sure that the constraints are met. This is known to be “good enough” solution. This process is prone to errors, inefficiency and violation of constraints especially when working in a highly tight scenario. Replicating the human way of solving the problem (Greedy algorithm) in computers is extremely inefficient due to its computing cost.

Computers are rarely used for solving this problem because of the stigma that it is unreliable and inefficient. However, this is back when computers are slow and not intended to solve compute intensive problems. With the explosion of big data and exponential increase in computing power, using computers to solve new sets problems that previously can only be solved by humans became easy. Scheduling is one of the problems that can now be solved by computers with solutions acceptable or even better. Artificial intelligence has become more common today brought by the growth in computing industry. Several solutions fall under artificial intelligence such as machine learning, deep learning and genetic algorithms. All of these processes have their own pros and cons. Genetic algorithm is fit for scheduling problem as it creates a solution over time based on rules and criteria.

In AMA Computer College Las Piñas, the college dean is responsible for creating schedules. The scheduling is done semi-manually with computer assistance. Multiple problems occur from this process. The usability and friendliness of the software that assists in scheduling is a

burden due to a number of features that are scattered and not even being utilized majority of the time. The process can still not provide a solution that is ensured to be free from errors, bad looking result and constraint violations. Though it has become a practice for the dean to make schedules, problems in making scheduling solution is not gone. Since the approval of K-12 education system in the Philippines, the institute has admitted senior high school students which put the dean in creating a schedule in a very tight condition. With the current conditions, using a computer to help in solving the scheduling problem can be an option. This research aims to create a candidate software solution by using genetic algorithms to generate schedule.

Statement of the Problem

Scheduling in AMA Computer College Las Piñas is being done semi-manually with computer assistance which can produce errors, violations of constraints and inefficiencies noting that scheduling has gone harder with tighter constraint. The problem can be broken down into;

- Scheduling is still being done semi-manually with the operator using the process alike to greedy algorithm which can produce results with errors.
- The software assisting the operator is not user friendly and the experience of using the software is bad and far from standards. This leads to confusion and inability to exert the software's full potential.

- The software being used is not fitted for the institute and therefore creates a steep learning curve. The universality of the application causes confusion and inefficiencies with preparation of scenario for generation of solution.

Significance of the Study

For AMA Computer College Las Piñas: The system will greatly reduce time and effort spent on generating a schedule, therefore giving more time for the administrator to manage the institute. This will also be applicable with the senior high school department of the said institute.

For AMA Education System and Other Educational Institutes: Multiple educational institutes can adopt the system and benefit by having a computed scheduling solution. It will help ease the management of school and possibly reduce expenses by having a more utilized schedule. The system is built to support scheduling for levels lower than college.

For the Researcher: Developing the system will greatly help the researcher towards improving on computer science due to implementation of multiple algorithms. It will also help provide a step for the growing artificial intelligence community. The research is going to be included in the portfolio which can display the skills of the researcher in tackling complex problems.

For Future Researchers: Researchers can use this research as a source for their research. They may use this approach on solving optimization problems or even create a more efficient version. This research comes with helpful data on understanding the performance and process which will help future researchers.

Conceptual Framework

The program can easily be described as a scheduling computing software wherein you input basic data sets and it will output a structured result. A simple input-process-model would be:

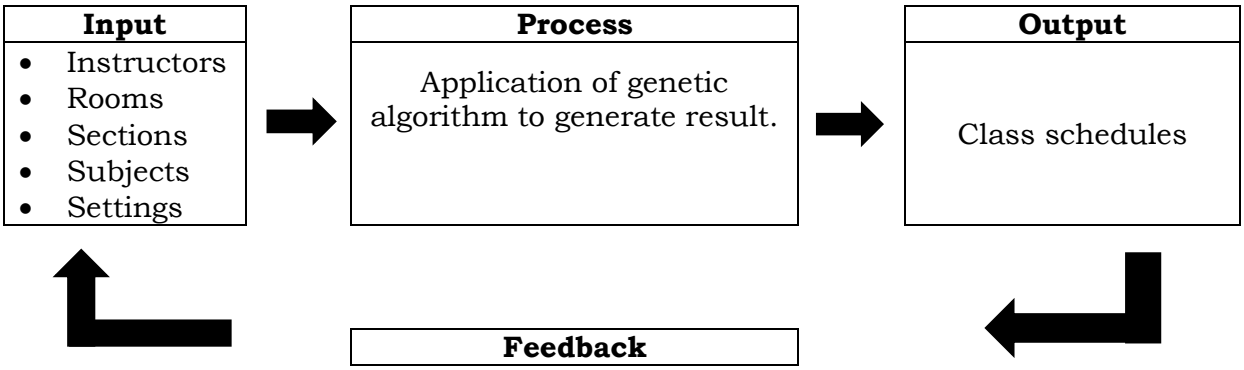

Figure 1.0 Simplified IPO-Model of the Program

Genetic algorithm is a computing method that natural selection which is a process in biological evolution. The process is repetitive and best conceptualized using a flowchart. In genetic algorithm, a solution is referred as chromosome instead of individual. The figure below shows the basic process of genetic algorithm. However, in practice of the system,

another step is added after evaluation which is adjustment of environment.

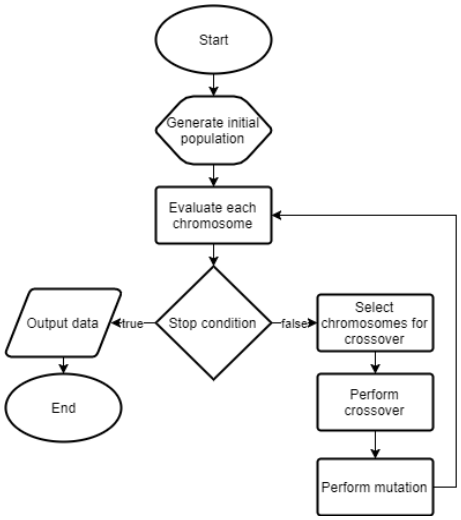

Figure 2.0 Basic Genetic Algorithm Process

Generation of population is to be done randomly with a mix of greedy approach where a random point will be selected and act as starting point and then start filling up the table with any entity that would fit. During generation of population, all hard constraints are to be met. Medium constraints will undergo multiple attempts to be followed while soft constraints are totally ignored.

Evaluation is the process of calculating the fitness of a chromosome. Fitness is the measure of quality of the chromosome relative to the desired solution. It also triggers the ending process given that a certain fitness is met. Calculating the fitness is done by assessing each chromosome to the selected medium and soft constraints specifically the

subject placement, lunch break, section rest, section idle time, instructor rest, instructor load balance and meeting pattern.

The fitness calculation would be largely based on the amount of plotted subject compared to the required subjects. However, the calculation would still depend on the evaluation matrix provided by the user. The evaluation matrix is a set of constraint weights that has the capability to shape the solutions. It is a list of containing the prioritization of constraints using a distribution of a hundred percent (100%)

Table 1.0 Evaluation Matrix

Name	**Function**
Subject Placement	Attempts to create valid schedule for all subjects of all sections.
Lunch Break	Attempts to at least have a 30-minute break between 11:00 AM and 1:00 PM
Section Rest	Attempts to at least give sections a 30-minute break for every 3 consecutive hours of session
Section Idle Time	Attempts to ensure that sections will have little or no idle time with respect to lunch break and section rest*. **Section rest is modified by removing the need for consecutive session.*

Instructor Rest	Attempts to at least give instructors a 30-minute break for every 3 consecutive hours of session
Instructor Load Balance	Attempts to balance the distribution of instructor load with respect to their pooled subjects.
Meeting Pattern	Attempts to ensure that all divided schedules are following a pattern instead of random date selection.

Computing the total fitness would use the formula below where:

x

$= chromosome, functions\ follow\ the\ evaluation\ matrix\ below\ and\ fW\ is\ evaluation\ weight$

$$\boldsymbol{fitness}(\boldsymbol{x}) = (g(x) * gW) + (h(x) * hW) + (i(x) * iW) + (j(x) * jW) + (k(x)\ kW) + (l(x) * lW) + (m(x) * mW)$$

Subject Placement

This evaluation stage calculates fitness of the chromosome base on the subject placement. Computing this would use the formula below where:

$$\boldsymbol{fitness} = \frac{amount\ of\ placed\ subjects}{total\ amount\ of\ committed\ subjects} * 100$$

Lunch Break

This evaluation stage checks if sections has lunch break all throughout the week. Attending section days are evaluated and tagged *"without lunch break"* if all timeslot between 11:00 AM to 1:00 PM is filled. Calculating the fitness follows the formula below:

$$\boldsymbol{fitness} = \frac{days\ without\ lunch\ break}{total\ amount\ of\ attending\ section\ days} * 100$$

Section Rest

This evaluation stage calculates the rest period of sections which is dictated by the rule that there should be at least a break for every three (3) consecutive hours of session. Calculating the fitness follows the formula below:

$$\boldsymbol{fitness} = \frac{days\ without\ section\ rest}{total\ amount\ of\ attending\ section\ days} * 100$$

Section Idle Time

This evaluation stage checks for the unnecessary gap between subject to subject of sections. The section idle time makes use of modified section rest evaluation. The requirement for consecutive session is removed, therefore gives a section free 30 minutes of rest time for every three hours of session present in the day. This stage also respects lunch

break and section rest. Therefore, idle time is computed using the formula below:

$$\boldsymbol{idle\ time}$$
$$= number\ of\ vacant\ timeslots\ between\ first\ and\ last\ session\ of\ the\ day$$
$$- allowed\ breaks$$

If idle time is more than zero for the day, then it should be tagged with excess rest. We can therefore compute the fitness of section idle time by:

$$\boldsymbol{fitness} = \frac{days\ without\ excess\ rest}{total\ amount\ of\ attending\ section\ days} * 100$$

Instructor Rest

This evaluation stage calculates the rest period of instructors which is dictated by the rule that there should be at least a break for every three (3) consecutive hours of session. Calculating the fitness follows the formula below:

$$\boldsymbol{fitness} = \frac{instructor\ days\ without\ rest}{total\ amount\ of\ teaching\ instructor\ days} * 100$$

Instructor Load Balance

This evaluation stage computes the normalized distribution of instructor load. This takes into account the fact that instructors may have a mutual subject pool with other instructors. The following formula below shows the formula for calculating the load balance:

$$\boldsymbol{instructor\ load}(\boldsymbol{instructor}) = \frac{amount\ of\ load\ taken\ by\ instructor}{total\ amount\ of\ load\ in\ the\ subject\ pool\ of\ instructor}$$

$$\boldsymbol{fitness} = \frac{\sum instructor\ load(instructors)}{total\ number\ of\ active\ instructors}$$

Meeting Pattern

This evaluation stage checks for the correction in the usage of meeting patterns. There are only currently two (2) defined meeting patterns which is "MWF" and "TTH". Calculating the fitness follows the formula below:

$$\boldsymbol{fitness} = \frac{amount\ of\ subjects\ abiding\ meeting\ pattern}{total\ amount\ of\ committed\ subjects - subjects\ indivisible} * 100$$

After evaluation, an additional step is added in order to perform the adaption of artificial intelligence to its current performance. Adaption is the process of changing running variables to cater for better results. This helps the artificial intelligence to avoid focusing on one set of problem which may cause pre-mature convergence. Adaption performs two important task which is population alignment and mutation rate adjustment.

Population Alignment

Population is the term for the group of solutions. Before generation of solution, the operator using the system can change the running variables of the algorithm. The settings “Minimum Population” and “Maximum Population” defines the limit of population change. Calculation of population alignment will show how much change in population is to be done using a series of formula below:

$$\boldsymbol{change}(\boldsymbol{x}) = \frac{\left(\frac{\sum instances\ of\ x\ with\ (-1 < deviation < 1)}{population\ count} * 100\right) - low\ variety\ settings}{100} * population\ count$$

Mutation Rate Adjustment

Mutation rate is the chance for each chromosome to get random change. Mutation rate adjustment happens when the trigger which is set before running is met. When a generation completes without triggering the adjustment, the mutation rate will decrement by 0.5% else it will be increased by 0.5%. Calculation of mutation rate adjustment trigger follows the formula below:

$$\boldsymbol{change} = 0.5 * (-1\ if\ average\ fitness - previous\ average\ fitness < adjustment\ trigger\ settings\ else\ 1)$$

After evaluation, chromosomes are picked to participate for reproduction. Chromosomes with higher fitness is more likely to get picked. This is where the survival of the fittest comes. There are multiple ways of selecting chromosomes all of which have their pros and cons. Having the most appropriate type of selection will help avoid early convergence and promote diverse solutions. As an elitist variant of genetic algorithm the top ***n%*** of population is guaranteed to proceed in the next generation with same genes depending on the settings.

The rest of the selection process is to be done by implementing multiple tournaments on the population. The participants in the tournament are selected randomly with a quantity of 4% of existing population but capped at twenty five (25). The tournament will run until enough pairs are picked. The following example below shows a simulation where 40% of the 10 chromosomes are selected in the tournament.

Table 2.0 Tournament Selection Example

Chromosomes	**Fitness**	**Participant**
#1	**4.1**	**TRUE**
#2	3.4	
#3	**5.6**	**TRUE**
#4	7.5	
#5	6.5	
#6	1.2	
#7	**3.5**	**TRUE**
#8	**2.8**	**TRUE**

#9 #10	7.5 1.1	
Winner: Chromosome #3		

If there are 100 chromosomes in a population with a 10% elitism rate, then we have to generate tournament 90 times in order to pair 45 chromosomes that will produce 90 offspring. The offspring and elite chromosomes after crossover would total into 100 which is the population count all throughout the process.

The mating pool consist of selected chromosomes that will undergo crossover. The crossover operation involves production of offspring based on genes of parents analogous to its biological counterpart. There are multiple ways to perform crossover and like selection, it is important to use the most appropriate one to have better outcome. Davis' order crossover works better for permutation based problems. The process works by taking the gene cluster of first parent into the offspring and then wrapping the gene cluster of second parent into the offspring. For each pair of chromosomes, two offspring will be produced where parent's role is reversed to produce different offspring. All offspring will take their parent's place in the population. This can be visualized below where:

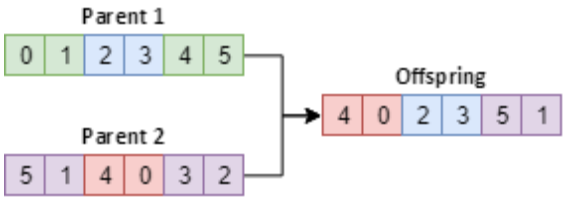

Figure 3.0 Davis' Order Crossover

Mutation is a genetic operator that helps maintain the variety of solutions. It can also prevent solutions from being stuck in local optima. Mutation is the process of altering the gene/s of a chromosome. Usually, mutation rate is low and fixed. However, adaptive genetic algorithm means that mutation rate may vary depending on the performance of the population.

Once the genetic algorithm terminates either by force or meeting the end criteria, the top 5 fittest chromosomes will be displayed for user review. The user may pick any from the proposed solution and get its output in a CSV file.

Scope and Limitation

This research aims to create an artificial intelligence that can create timetable schedules. This only covers processing input data and generating human sensible results. The process to be used in creating schedules is adaptive-elitist genetic algorithm. A type of evolutionary algorithm that involves keeping the best set of solutions over next generations to preserve high level solution but at the same time, variables such as the population count, mutation rate and selection pressure, change in order to avoid premature convergence that leads to poor output.

Computing the schedule would involve validation for each constraint. Constraints are set of rules that directs the acceptance of the output. There will be three types of constraints to be used;

1. **Soft Constraints** – A set of rules that can be broken without affecting the validity of the output.
2. **Medium Constraints** – A set of rules that can be broken with an effect to the validity of the output. However, this can only be broken if the scenario is logically invalid or impossible.
3. **Hard Constraints** – A set of rules that would produce an invalid solution if broken.

Table 3.0 Constraints

Hard Constraint	**Medium Constraint**	**Soft Constraint**
Instructors teach one class at a time	Sections' subjects are placed on the schedule	Students should have only 30 minutes break for every two hours of session per day
Instructors teach at their available schedule	Sections should have at least 30 minutes vacant time between 11:00 AM to 1:00 PM for lunch break (Optional)	Subjects that are divided should follow the two defined meeting pattern which is "MWF" and "TTH"

Instructors can only take *N* amount of subjects dependent to their maximum amount of load	Sections should have at least 30 minutes vacant time for every 3 consecutive hours of session	Instructors should have normalized distributed load based on the instructor pool of subjects
Sections attend one class at a time	Instructors should have at least 30 minutes vacant time for every 3 consecutive hours of session	
Sections attend at their available schedule		
Sections will stay at one room unless taking a laboratory subject		
Sections with subjects that are shared to other sections should produce a schedule that is compatible to all sharing participants		

Rooms can only take class at its available schedule Subjects have a right room type Subjects will divide depending on its configuration		
Subjects will be assigned an instructor if there is a pool of instruction on its configuration		

The application takes five major input; Instructors, rooms and subjects component supports addition of entry using the application and importation. The sections component allows creation of sections with special feature of sharing subject with other sections and the scenario manager component handles running configuration. There would be two types of output; using the result viewer of the application which can view the top five solutions of the last generated scenario and exporting the selected solution which will produce three comma separated values (CSV) files. Tests for reliability and performance of the program specifically the evaluation for the performance of the artificial intelligence will come from usage of application using actual records of AMA Computer College Las Piñas. The application will be using the Python language. The research will

be conducted during the second and third trimester of the school year 2017-2018 in AMA Computer College Las Piñas.

Results of the application is not guaranteed to be the best possible solution for the scenario. The quality of the result relies heavily on running preference. Due to its stochastic nature, results may vary from poor to excellent. The system does not guarantee that every solution's hard constraint will be met especially when the presented scenario is logically impossible or intensely tight. This also means that when in a strict scenario soft or medium constraints are subject to violation in order to meet hard constraints. Hard constraints are bound to be followed and will never be violated. Importation of values from CSV supports strict formatting shown below.

CSV Formatting

All CSV files should use line 1 of the file as file indicator. These should be one of the following "instructors", "rooms" and "subjects". The second line of the file should be used for defining table columns in same order with the writing below:

1. **Instructors** – name, hours

Table 4.1 Instructors CSV Columns

Column	**Description**	**Type (Options)**
name	Name of the instructor	String
hours	Serviceable hours per week	Integer (1-100)

2. **Rooms –** name, type

Table 4.2 Rooms CSV Columns

Column	**Description**	**Type**
name	Name of the room	String
type	Type of the room	String (lec, lab)

3. **Subjects –** code, title, type, hours, splittable

Table 4.3 Subjects CSV Columns

Column	**Description**	**Type**
code	Short code of the subject	String
title	Name of the subject	Integer (1-100)
type	Type of the subject	String (any, lec, lab)
hours	Hours per week of the subject	Double ***+.5*** (1.0, 12.0)
splittable	If the subject is going to be splitted	Boolean-Integer (0-1)

Example Importable CSV File

first_trimester_subjects.csv

```
subjects
code,title,type,hours,splittable
ENGL101,Communication Skills 1,lec,3,1
MATH101,College Algebra,lec,3,1
MATH110,College Trigonometry,lec,3,1
CS101,Introduction to Computing,lec,2,1
CS101L,Introduction to Computing,lab,2,0
```

The application offers a way to customize the running configuration for genetic algorithm which can cause performance issues and improper result if tweaked wrongly. Computers that will run the system should follow the minimum required specifications shown below. The system is not liable for any 3rd party damages that may occur due to inappropriate use.

Table 5.0 Minimum System Requirements

Minimum System Requirements	
Operating System:	Windows 7 and Above
CPU:	2.0+ GHz with multithreading support
RAM:	1 GB* **This still relies on the scenario size and algorithm configuration see Chapter 5 for assessment of consumption.*
Disk Space:	At least 1 GB Available Space* **User generated files are excluded in assessment and minimum space is also subject to application usage.*

Genetic Algorithm Settings

The configuration of genetic algorithm within the application is not one for all type where it can cater every scenario. This is due to the limitation of computational power. To counter the limitation and still be able to support scalability of usage, the algorithm's running configuration can be altered according to operators' device capability or preference. The following table below shows the description and recommendation for each modifiable setting property of the algorithm.

Table 6.0 Genetic Algorithm Settings

Settings	**Description**	**Limit**
Minimum Population Count	The minimum and starting number of chromosomes in a generation. Fifty (50) is the suggested value for this as it is high enough for a variety of solution.	50-10,000
Population Count	The limit on how many chromosomes can a generation have. Calculating how much the computing device can handle depends on its RAM. For every seventy (70) active entities in a scenario (subject, instructor, room, and section), it will consume one megabyte (1 MB) of memory per chromosome. Therefore, if there are using fifty (50) entities and maximum population count of one	50-10,000

	hundred (100), the peak consumption of RAM would approximate to 72 MB plus 25% of the value for main application handling. A low-end computer should maintain low maximum population count, it can match the minimum population suggestion of fifty (50).	
Maximum Generations	The limit of how many generations the algorithm can produce. Upon reaching the set value, the algorithm will stop. The higher entity count or the more complex constraints the scenario presents, the higher this should be valued as it gives more chance for the artificial intelligence to fix minor problems or even create a breakthrough that can change the entire solution set.	50-10,000
Maximum Creation Attempts	The maximum amount of attempts for the artificial intelligence to place a valid subject schedule in the chromosome. A strong computer can put higher value on this as this serves more as a brute force tactic which can help increase the quality of the solution.	1,500-30,000

Mutation Rate Adjustment Trigger	The threshold value for triggering mutation rate adjustment. The value checks for the difference of the latest average fitness to its previous average fitness. If the difference falls below or equals to the set value, the mutation rate will increase by 5% while reducing it by 5% if not. The value can be set to zero if the operator does not want to change the existing mutation rate of 10%. It is recommended to keep the value of this setting to its default of 0.08 to prevent massive changes on the population.	0.00-100.00
Maximum Fitness	The trigger for ending the algorithm. Once a chromosome hits the set value, the algorithm will stop. If a perfect solution based on model is desired, this can be set to 100%. However, the chance of getting a perfect solution on every combination of model is not guaranteed. It is suggested to place an optimal value for this which can range from 90-98% for acceptable level.	0-100%
Elite Population	The percent of how much of the current population's best performing	0-100%

	chromosomes will be preserved until the next generation. The suggested value for this is 5% as it keeps a lower count of good chromosomes and helps prevent premature convergence.	
Deviation Tolerance	Measures the richness of the population's solution set. Each chromosome's fitness is measured and normalized for statistical deviation. If one of the deviation sigma goes beyond the set value, the population count changes by how much percent it is off else the population will be reduced by how much percent it lacks. This helps in maintaining the variety of solutions basing on the deviation of fitness between chromosomes. The suggested value is 55% as it prevents total domination of a common group of chromosomes.	0-100%

The suggested values do not guarantee the best combination. The computation of space consumption does not fully depict the actual usage.

Definition of Terms

Operational Terms:

Chromosome - A set of parameters which defines a proposed solution that the genetic algorithm is trying to solve.

Crossover - A genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based.

Deep Learning - A part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, partially supervised or unsupervised.

Gene - The basic unit of a chromosome. It represents as a part of a solution.

Genetic Algorithm - A metaheuristic inspired by the process of natural selection that belongs to the larger class of evolutionary algorithms (EA). Genetic algorithms are commonly used to generate high-quality solutions to optimization and search problems by relying on bio-inspired operators such as mutation, crossover and selection.

Genotype - The genetic representation of the solution, in a basic solution, it might be represented as a string of binary.

Machine Learning - A field of computer science that gives computers the ability to learn without being explicitly programmed.

Mutation - A genetic operator used to maintain genetic diversity from one generation of a population of genetic algorithm chromosomes to the next. It is analogous to biological mutation.

NP-Complete - A complexity class where solutions can be verified quickly and if there is a quick algorithm to solve this type of problem, it can be used to solve other NP problems too.

Phenotype - The real world representation of the solution.

Population - A set of proposed solutions.

Selection – Stage of GA that selects chromosome for future combinations.

Technical Terms:

Chromosome - A DNA molecule with part or all of the genetic material of an organism.

Crossover - A point or place of crossing from one side to the other.

Gene – Sequence of DNA which codes for a molecule that has a function.

Genotype – The part of the genetic makeup of a cell.

Mutation – The permanent alteration of nucleotide sequence of the genome of an organism.

Phenotype – The composite of an organism's observable characteristics or traits.

Population – All the inhabitants of a particular location.

Selection – The action of carefully choosing someone as being the best.

CHAPTER 2

REVIEW OF RELATED LITERATURE AND STUDIES

University timetable scheduling is a combinatorial problem where finding the best solution is hard when the search space goes big. Knowing the background of the problem helps understand why even at computing era, solving it is a somewhat monumental task. Getting to know the possible solutions for the problem is also needed as it might show some of the shared traits that might lead to a breakthrough in finding a better solution. Finally, gathering information regarding the proposed solution will help guide the research into forming better solution.

1. *Operations Research*

The article published in Universal Class [1] and in the chapter 14 of the book published in IBS center for management research [2] it is stated that operations management is a delicate part of business which is responsible for the process and quality of work. It is said that scheduling plays a major role in meeting the customer's requirements on time and improving the efficiency of the overall system. In a report by R. Dan Reid and Nada R. Sanders [3], they said that the theory of constraints is that the result of a schedule is determined by the constraints. In order to avoid getting overwhelmed by the constraints, one must identify bottlenecks in the process and try to improve it.

2. *Common Scheduling Conditions in the Philippines*

In a university-based news article published by Althea Gonzales and Jessy Go [4], students have expressed mixed reactions on their university's newly imposed scheduling rule. The classes were made earlier and a break of fifteen (15) minutes were given between each subject. It is said that it was implemented in order to give students time to prepare for their classes and avoid missing out. Educational institutes sometimes implement their own scheduling schemes in order to compensate to their situations. In a recent independent news assessment of K-12's status in the Philippines written by Anne Marxze Umil [5], many institutes still lacks in resources needed to accommodate students. There are still shortages in classrooms and teachers. This also includes private schools who had to receive a sudden influx of students supported by the voucher system. In order to solve the problem on lack of resources, according to an article written by Careen L. Malahay and Tweeny M. Malinao [6], schools have implemented a shorter scheduling scheme to create shifts in order to accommodate the students. The effects of shortages in resources can also be felt in college level. In the article published by Paulyn Navarrete and Illiana Tan, it has showed the dismay of students who cannot create a comfortable schedule due to few subject openings and conflicts. This is common among institutes who of course cannot handle every scenario.

3. *How Humans Solve Complex Problems: The Case of the Knapsack Problem*

According to an article in University of Colorado [8], knapsack problem is one of the classic problems in the field where computing schedules belong. It is a combinatorial optimization problem with the goal of finding the subset of items with the highest total value from a set of items with given weight and value subject to weight constraint. In the research about how humans solve complex problems written by Carsten Murawski and Peter Bossaerts, it is found that humans solve the knapsack problem by implementing a greedy algorithm process of thinking in the early to middle phase of solving the problem followed by a branch and bound like searching technique. In the experiment where the average number of full knapsack configuration is one thousand three hundred eighty one (1,381), the chance of getting the most optimal solution is merely a 0.7%. The results of the study have shown that humans performed extremely well with a median score of 97.4% of the most optimal solution value. However, it is worth noting that just because humans can solve knapsack problems with an optimal solution, it is not the same case with scheduling problems.

4. *NP-Complete Complexity Class*

The knapsack problem belongs to the complexity class of non-deterministic polynomial-time (NP) hard. These are problems where there is no efficient solution algorithms yet. The computational time only grows as the size grows according to Stephen A. Cook [10]. According to standard

defining organization's computer scientist, Paul E. Black [11], NP-Complete is a complexity class where solutions can be verified quickly and if there is a quick algorithm to solve this class of problem, it can be used to solve other NP problems too. In an article by J.D. Ullman [12], he said that finding the most optimal schedule or solving scheduling itself belongs to NP-Complete problem.

5. *Ways on Solving University Class Scheduling*

University class scheduling problem belongs to optimization problems in operational research. It is considered as one of the most difficult problems faced by universities and colleges today according to a study on subject matter by Ahmed Wasfy and Fadi A. Aloul [13]. There are multiple approaches towards solving scheduling problems according to Ellis Cohen and Jarurat Ousingsawat [14]. Some are only feasible up to certain level such as the constraints they can handle, objectives to meet and computation time. The most common approaches are construction algorithms which greedy algorithm is included and search algorithms where tabu search, simulated annealing, particle swarm optimization, genetic algorithm and many more from J Zhou, P E D Love, X Wang, K L Teo and Z Irani's review on scheduling [15]. Construction algorithms can and sometimes mixed with search algorithms such as greedy and genetic algorithm where the population may depend on the generated individuals by the greedy algorithm.

6. *Genetic Algorithms: An Overview*

According to Melanie Mitchell's lesson [16], genetic algorithm is a program that mimics the process of biological evolution in order to solve problems and to model evolutionary systems. The vast search space has inspired the creation of genetic algorithm. Biological entities over the years have adapted to its environment to be more efficient and effective. It has then become part of the early waves of artificial intelligence wherein rules and behavior are manually encoded. The algorithm is still in use in modern period mostly for optimization problems.

7. Phenotypes, Genotypes, and Operators in Evolutionary Computing

In David B. Fogel's publication [17], genetic algorithm works by evolving a set of solutions into better ones. Each solutions are often referred as chromosome. A group of chromosomes creates a population. A solution is represented into two types, genotype and phenotype. The genotype is the genetic representation of the solution, in a basic solution, it might be represented as a string of binary. The phenotype is the real world representation of the solution. In evolutionary computation, it is important to convert phenotype into genotype as it is easier to compute. Having a good and proper conversion will definitely help in computing the solution.

8. *Selection Methods for Genetic Algorithms*

The process of genetic algorithm starts with initialization of population. After initialization, evaluation of chromosomes are done to give

score on how did each solution perform. In order to continue into next generation of solutions, selection operation is done. Selection is the process of selecting chromosomes to be used as parents to produce next generation of solutions. There are multiple types of selection which can be used to support multiple types of problem. Some of these are roulette wheel selection, stochastic universal sampling, linear rank selection, exponential rank selection, tournament selection and many more according to Khalid Jebari [18]. Tournament selection is done by selecting random chromosomes wherein the best performing will be selected. This will be done until enough chromosomes are selected. The performance of tournament selection is incredibly fast but depends on its configuration. It can either lead to fast good or bad convergence. However, the selection pressure is high due to the amount of repetitive computation.

9. Crossover Operators in Genetic Algorithms: A Review

Crossover is a genetic operator which plays one of the biggest role in genetic algorithm. It is done by creating an offspring based on the genes of the parents mimicking the biological reproduction. There are many ways on how to perform crossover but just like selection, each of them has their own pros and cons. Some of these crossover types are standard crossover using N-points, shuffle, discrete, uniform, order based and many more according to A.J. Umbarkar and P.D. Sheth [19]. Order based crossover is most appropriate for problems where groups or patterns are important. It helps preserve existing combination while improving trying to improve it.

10. Genetic Algorithm Essentials, Studies in Computational Intelligence

The final operator in genetic algorithm is mutation. Mutation is a random based operator that changes some gene in a chromosome. It helps prevent early convergence by introducing random changes that might increase or decrease the fitness of a chromosome. There are three requirements in mutation; reachability, unbiasedness and scalability according to O. Kramer's study [20]. Usually, a fixed value for mutation rate is used but adaptive variety of genetic algorithm may change the value from generation to generation depending on the situation.

Scheduling is one of the most important part of a business. It plays a big role in operations management as it helps maximize the efficiency of production or service. In educational institutes, scheduling is one of the biggest challenge often because bad schedules leads to students and other party's dismay. Solving the problem using only pure man work maybe efficient enough but it will not scale up. To solve this problem, computing the schedule using the right algorithm and constraints is a must. Genetic algorithm presents itself as a valuable search optimization solution with its capability to handle and manipulate data.

11. User Experience for Windows Desktop Applications

Microsoft's Windows Development Center has created design principles [20] for Windows desktop applications. Some of the important considerations are categorized below:

- **Windows** – The application should be able to support 800x600 pixels resolution. It should be displayed initially on the center of the user's window.
- **Layout** – The size of controls and panes within a window match their typical content.
- **Text** – Use ordinary, conversational terms and use title-style capitalization for titles while sentence-case for all other UI elements.
- **Controls** – Add label to controls that requires user input. Use tooltips for unfamiliar features.
- **Accessibility** – The operator should be able to navigate the full window using tab key.
- **Dialogs** – The dialog should present important data and interaction.

CHAPTER 3

RESEARCH DESIGN AND METHODOLOGY

Artificial intelligence is a computer science field that is being used mostly on problems that includes precision or optimization. The research falls under optimization problem and evaluation of performance in an extensive verification is not a feasible solution. It is important to select an appropriate research design and methodology to get the viability of the research.

Research Design Methodology

The selected method for collecting and analyzing data for testing the system is quantitative. The data to be collected are sets of school configuration (e.g. subjects, instructors and rooms). These datasets will be profiled after being used in the system. The profiling results shall determine the performance of the system by performing artificial intelligence systems based evaluation.

Method of Software Development

Iterative development as a method for creating the software fits for continuously changing environment for artificial intelligence development. This enables the developer to learn every iteration and apply it to future instance. This methodology is fit for the research as the system relies on

tweaking and managing constraints. Each iteration is an enhancement to the system which will repeat until the system delivers its expected result.

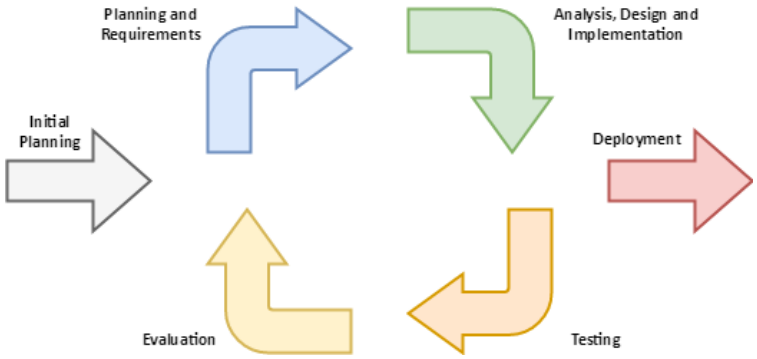

Figure 4.0 Iterative Development Methodology

Initial planning has been done by designing system architecture, data schematic and models. Upon entering an iteration, the feature to be done is decided and planned. There are two types of iterations used; adding a feature or a model and tweaking values. Each iteration's goal is then implemented and then manually tested.

Method of Software Evaluation

The selected evaluation methodology for the research is a combination of Monte Carlo methods and surrogate modelling. It is an experimental methodology as evaluation for evolutionary computation based scheduling systems are yet to be done.

The Monte Carlo methods are a class of computational algorithms that uses repeated random sampling to obtain numerical results. It is often used under the optimization and probability problem sets. It works by tweaking the initial value and trying to produce a solution from it.

The surrogate modelling is often used in science and engineering field where the outcome cannot be easily measured directly so a model is used instead. An example of this is finding the optimal design for a car's aerodynamics. Computing the design's performance with real life like simulations is expensive so instead, an emulator model is used. This generally works by creating an approximation of models which mimics the behavior of real model as closely as possible without the need for a more expensive evaluation.

Implementation of Monte Carlo method and surrogate modelling for evaluation of artificial intelligence based scheduling system, specifically the researcher's study shall follow the following steps:

1. Generation of a set of solutions using Monte Carlo method with genetic algorithm settings as variables that can be tweaked.
2. Creation of a surrogate model which only uses the evaluation matrix as basis.
3. Computation of each set of solutions generated by the Monte Carlo method difference to surrogate model.
4. Providing the mean of the result as basis for performance of the artificial intelligence (Scenario based).

Usage of Monte Carlo method in system is for generation of random genetic algorithm configuration. The surrogate modelling usage for this research makes use of an imaginary 1-dimensional metric basing on the

evaluation matrix. The model will serve as a basis for evaluating the closeness of the solution to the ideal one. In conjunction with software evaluation methodology using the proposed steps above, data gathering would then lean towards data-centric approach. This is to provide correct conclusions towards the conclusion of the research. The proposed data to be gathered are the ones to be used by the evaluation above which will be composed of mostly result performance grade (fitness).

CHAPTER 4

PRESENTATION, ANALYSIS AND INTERPRETATION OF DATA

Using the surrogate modelling and combining it with Monte Carlo method has presented multiple data for interpretation. The following figure below shows the chain of data collection and evaluation methodology for one model:

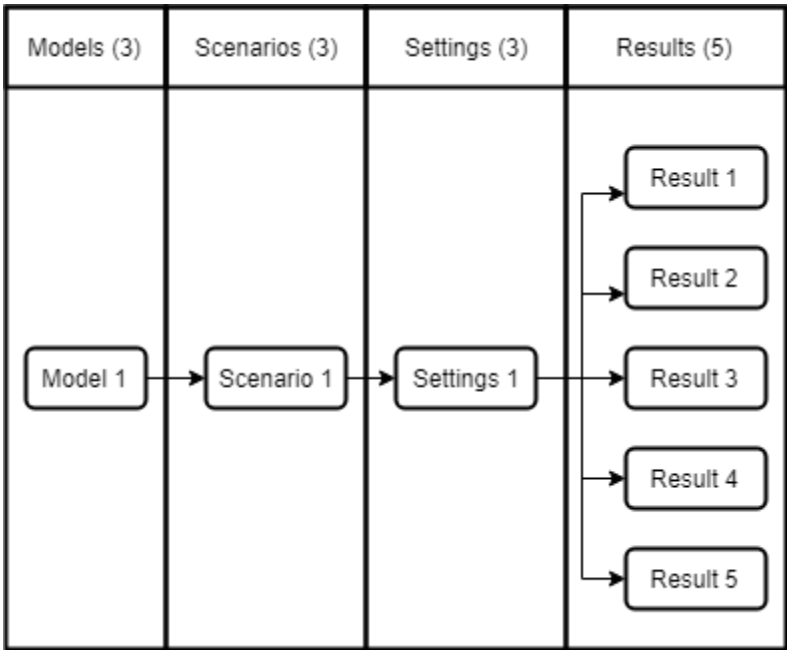

Figure 5.0 Evaluation Tree (Simplified Branch)

There will be three (3) models to be used listed on the table below. For each model, there will be three (3) scenarios. For each scenario, there will be three (3) randomly generated value of settings. It is important to note that there will only be one time generation of random values for settings and the three (3) generated values will be used all throughout the scenarios to prevent bias in result. For every setting, there will be five (5)

generated result set. In total, there will be 135 solution set for evaluation which will be abstracted per level.

Table 7.0 Selected Evaluation Matrix

Name	**Matrix**	
Balanced Distribution	Subject Placement	16%
	Lunch Break	14%
	Section Rest	14%
	Section Idle Time	14%
	Instructor Rest	14%
	Instructor Load Balance	14%
	Meeting Pattern	14%
Subject Placement	Subject Placement	100%
	Lunch Break	0%
	Section Rest	0%
	Section Idle Time	0%
	Instructor Rest	0%
	Instructor Load Balance	0%
	Meeting Pattern	0%
Tight Constraints	Subject Placement	20%
	Lunch Break	0%
	Section Rest	20%
	Section Idle Time	20%
	Instructor Rest	20%
	Instructor Load Balance	20%
	Meeting Pattern	0%

The randomization has a set amount of possible values based on the computer's optimal capability and suggested values from genetic algorithm settings (*Table 6.0 Genetic Algorithm Settings*) which can be viewed in the table below:

Table 8.0 Random Settings Limitation

Name	**Limit**
Minimum Population Count	50-200
Maximum Population Count	Minimum Population Count-200
Maximum Generations	50-150
Maximum Creation Attempts	1,500-4,500
Mutation Rate Adjustment Trigger	0.00-1.00 where value is divided by five (5)
Maximum Fitness	90-100
Elite Population	0-10%
Deviation Tolerance	50-75%

Using randomization provided above, here are the three sets of genetic algorithm settings:

Table 9.0 Random Settings

Entry	**Matrix**	
#1	Minimum Population Count 63 Maximum Population Count 84	Mutation Rate Adjustment Trigger .09 Maximum Fitness 96%

	Maximum Generations	64	Elite Population	6%
	Maximum Creation Attempts	1,873	Deviation Tolerance	57%
#2	Minimum Population Count	79	Mutation Rate Adjustment Trigger	.05
	Maximum Population Count	148	Maximum Fitness	93%
	Maximum Generations	89	Elite Population	7%
	Maximum Creation Attempts	2,279	Deviation Tolerance	69%
#3	Minimum Population Count	78	Mutation Rate Adjustment Trigger	.11
	Maximum Population Count	96	Maximum Fitness	95%
	Maximum Generations	63	Elite Population	2%
	Maximum Creation Attempts	1,513	Deviation Tolerance	63%

There are three (3) scenarios prepared for the evaluation. The three scenarios are imitation of the curriculum for each trimesters on AMA Computer College Las Piñas have (*Appendix C*) for CS, IT and CpE courses. Each scenario have 91, 93 and 95 entities respectively. These entities are active instructors, rooms, subjects and sections. Each model below shows the summarized version of each scenario. The summarization includes computation of min, max and average of total fitness of each result from all settings.

Model 1: Balanced Distribution

Table 7.1 Balanced Distribution

Name	**Matrix**
Subject Placement	16%
Lunch Break	14%
Section Rest	14%
Section Idle Time	14%
Instructor Rest	14%
Instructor Load Balance	14%
Meeting Pattern	14%

Using the current model, no scenario solutions was able to reach the maximum possible solution. The distribution of priority has made the artificial intelligence have hard time balancing all the constraint. The table

below shows the performance of the algorithm to each scenario in combined settings evaluation.

Table 10.1 Model 1 Summary

Scenario	**Highest Fitness**	**Average Fitness**	**Lowest Fitness**	**Average Time**
1	84.45 (Settings 2)	80.32	78.61	21:28
2	84.50 (Settings 2)	81.01	78.94	23:30
3	84.48 (Settings 2)	80.67	79.0	22:48

Model 2: Subject Placement

Table 7.2 Subject Placement Evaluation Matrix

Name	**Matrix**
Subject Placement	100%
Lunch Break	0%
Section Rest	0%
Section Idle Time	0%
Instructor Rest	0%
Instructor Load Balance	0%
Meeting Pattern	0%

Being one of the easiest model, the artificial intelligence was able to adapt easily as it does not have to worry about other constraints when

placing entities. It has showed its superiority on initializing the population to provide acceptable solutions even though it is just starting.

Table 10.2 Model 2 Summary

Scenario	**Highest Fitness**	**Average Fitness**	**Lowest Fitness**	**Average Time**
1	100.00 (Settings 1)	99.69	97.15	4:29
2	100.00 (Settings 1)	99.16	93.46	5:19
3	100.00 (Settings 1)	99.69	97.15	5:16

Model 3: Tight Constraints

Table 7.3 Tight Constraints Evaluation Matrix

Name	**Matrix**
Subject Placement	20%
Lunch Break	0%
Section Rest	20%
Section Idle Time	20%
Instructor Rest	20%
Instructor Load Balance	20%
Meeting Pattern	0%

The third model which can be considered as closest to real schedule in terms of software capability was able to assess solutions that are much

better than the first model despite having the same nature of balance in constraint prioritization. This has shown the system was capable of creating solutions that does not violate too much constraint.

Table 10.3 Model 3 Summary

Scenario	**Highest Fitness**	**Average Fitness**	**Lowest Fitness**	**Average Time**
1	88.87 (Settings 2)	87.22	85.74	21:27
2	89.35 (Settings 2)	87.66	85.94	23:32
3	89.09 (Settings 2)	87.44	85.84	24:33

The models have shown that the artificial intelligence was able to cope up with the problem. Each settings has produced varying answer but have shown that from the data gathered, higher settings can produce higher quality results.

CHAPTER 5

SUMMARY, CONCLUSION AND RECOMMENDATION

Summary of Findings

The usage of models was meant to test the capabilities of the artificial intelligence. Judging by the performance of results compared to models, the system was able to generate solutions that have at least 80% fitness (basing on set of highest solutions per scenario). The problem lies on the inability of artificial intelligence to perform minor corrections on adjusting schedules (see generation results category "C"). Nevertheless, the algorithm was able to cater for majority of the entries on the evaluation matrix.

The lack of testing in models and the amount of model itself does not show the exact capabilities and limitation of the system. With generation of results from one (1) model, three (3) settings and one (1) scenario taking almost an hour, there is only a limited amount of data that can be used for evaluation. However, it is fair to say that the system has performed to its capability despite of limitations imposed by the algorithm's configuration.

It is important to note that the generated random values for settings was limited, therefore, the system was not able to perform at its peak. The

tremendous amount of possible combinations for generation of the system have made finding the actual evaluation elusive.

Conclusion

The results have shown that the system can provide valid solutions that can be used. However, it does not provide complete automation. There are still scenarios that would require the operator to adjust some entries to create a perfect solution.

The system was also designed to be simple and straightforward. This eliminates any confusion caused by scattered user interface controls and makes usage of the software fully utilized.

The simplicity of the system and introduction of configurable algorithm's goal and performance reduced the need for so much constraints as solutions are made dynamically. This enables users to easily use and experiment with the application until they find perfect fit for their scenario.

The large amount of combinations for testing in order to find an accurate evaluation for the application has proven to be far from possibility. However, it can be concluded that from the models provided, the system was able to generate results that despite being imperfect still

remains valid and acceptable given the number of constraints imposed to it.

The solutions that the system will provide will heavily depend on the running configuration and evaluation matrix. One may find a perfect solution if the application was given enough time and computing power. The complete evaluation for the system will remain hard to solve as the freedom for the configuration of the algorithm has provided a large amount of combinations. It can also be inferred that evaluation using other methodologies will yield the same amount of result.

Recommendation

The researcher's goal was to develop a solution for timetabling problem. However, this does not mean that it should be used for every case where more constraints are being implemented than the available ones in the system. The researcher strongly recommends usage of other timetabling solution for more complex and constraint needs until further development on the existing system is available. However, usage of the system is still promoted as it is easier to use than other solutions.

Bibliography

[1] "The Usage of Operations Management", Universal Class. Retrieved from https://www.universalclass.com/articles/business/the-usage-of-operations-management.htm

[2] "Operations Management: Chapter 14", IBS Center for Management Research. Retrieved from http://www.icmrindia.org/courseware/Operations%20Management%202nd%20Edition/Operations%20Scheduling.htm

[3] R. Dan Reid and Nada R. Sanders, "Operations Management: Chapter 15", 2010. Retrieved from www.csus.edu/indiv/b/blakeh/mgmt/documents/opm101chapter15_000.ppt

[4] Althea Gonzales and Jessy Go, "Earlier classes, longer breaks draw mixed reactions", June 2014. Retrieved from http://thelasallian.com/2014/06/10/earlier-classes-longer-breaks-draw-mixed-reactions/

[5] Anne Marxze Umil, "New school year, same old problems: K to 12, shortages in classrooms, teachers", June 2017. Retrieved from http://bulatlat.com/main/2017/06/06/new-school-year-old-problems-k-12-shortages-classrooms-teachers/

[6] Paulyn Navarrete and Illiana Tan, "Persistent issues faced during enlistment", January 2015. Retrieved from http://thelasallian.com/2015/01/29/in-review-persistent-issues-faced-during-enlistment/

[7] Careen L. Malahay and Tweeny M. Malinao, “Elementary school shortens classes to accommodate pupils”, June 2012. Retrieved from http://newsinfo.inquirer.net/207699/elementary-school-shortens-classes-to-accommodate-pupils

[8] “Knapsack Problem Algorithms”, University of Colorado, Denver, May 2017. Retrieved from http://math.ucdenver.edu/~sborgwardt/wiki/index.php/Knapsack_Problem_Algorithms

[9] Carsten Murawski and Peter Bossaerts, “How Humans Solve Complex Problems: The Case of the Knapsack Problem”, October 2016. Retrieved from https://www.nature.com/articles/srep34851/

[10] Stephen A. Cook, “An Overview of Computational Complexity”, June 1983. Retrieved from https://dl.acm.org/citation.cfm?doid=358141.358144

[11] Paul E. Black, “NP-Complete”, December 2016. Retrieved from https://xlinux.nist.gov/dads/HTML/npcomplete.html

[12] J.D. Ullman, “NP-Complete Scheduling Problems”, June 1975. Retrieved from http://www.sciencedirect.com/science/article/pii/S0022000075800080

[13] Ahmed Wasfy and Fadi A. Aloul, “Solving the University Class Scheduling Problem Using Advanced ILP Techniques”. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=83CF2A3380EB4E45C5B2241036C7FAE2?doi=10.1.1.140.3740&rep=rep1&type=pdf

[14] Ellis Cohen and Jarurat Ousingsawat, “An Introduction to Algorithms for Solving Schedule-Related Problems”, January 2015. Retrieved from http://www.project.net/introduction-algorithms-solving-schedule-related-problems

[15] J Zhou, P E D Love, X Wang, K L Teo and Z Irani, “A review of methods and algorithms for optimizing construction scheduling”, March 2013. Retrieved from https://link.springer.com/article/10.1057/jors.2012.174

[16] Melanie Mitchell, “Genetic Algorithms: An Overview”, 1995. Retrieved from http://ohm.ecce.admu.edu.ph/wiki/pub/Main/ResearchProjects/mitchell_GA_tutorial.pdf

[17] David B. Fogel, “Phenotypes, Genotypes, and Operators in Evolutionary Computing”. Retrieved from https://pdfs.semanticscholar.org/1db9/a59ebe9b4808722fee55601beb8e2eda5064.pdf

[18] Khalid Jebari, “Selection Methods for Genetic Algorithms”, December 2013. Retrieved from https://www.researchgate.net/publication/259461147_Selection_Methods_for_Genetic_Algorithms

[19] A.J. Umbarkar and P.D. Sheth, “Crossover Operators in Genetic Algorithms: A Review”, October 2015. Retrieved from http://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf

[20] O. Kramer, “Genetic Algorithm Essentials, Studies in Computational Intelligence Chapter 2”, 2017. Retrieved from www.springer.com/cda/content/document/cda_downloaddocument/9783319521558-c2.pdf

[21] Microsoft, “UX Checklist for desktop applications”. Retrieved from https://msdn.microsoft.com/library/windows/desktop/dn742479.aspx

Appendix A

Letters

Appendix B

Title Proposal

Appendix C

College Curriculums

Appendix D

Sample Output

Appendix E

Program Listing

Appendix F

Application Results

Appendix G

Gantt Chart

Appendix H

Curriculum Vitae

Appendix I

Final Permit