**Project ARGS: Automated Registration Guidance System**

| | Michael Bernard Bryan D. Doron  University of the Philippines  Los Banos  Los Banos, Laguna, Philippines  mddoron@up.edu.ph | Clarice Ann T. Tagarda  University of the Philippines  Los Banos  Los Banos, Laguna, Philippines  cttagarda@up.edu.ph | Toni-Jan Keith Monserrat  University of the Philippines  Los Banos  Los Banos, Laguna, Philippines | | --- | --- | --- | |  |  |
| --- | --- | --- | --- | --- | --- |

# ABSTRACT

Generating a Class Timetable or Schedule has been one of the problems encountered during the registration process. Currently, the University of the Philippines Los Banos uses REGIST, an automated system designed to generate pre-enlisted student timetables. However, it has been inconsistent in generating initial timetable or schedules to students. ARGS, aims to improve on the current implementation of REGIST by using heuristic search algorithms such as Backtracking and Recursive Algorithms. It targets to increase the number of timetables with complete units, and ensure the quality of subjects being generated by ensuring that Major subjects with higher priority will be enlisted first before General Education (GE) subjects. The system will be compared to REGIST in terms of the number of students who obtained a complete timetable or complete number of units, the number of students who obtained the adequate schedule based on their current priority, and the total number of schedules generated: per year, per curriculum, and in a general sense.

# INTRODUCTION

Class timetabling or scheduling is one of the problems that the University of the Philippines Los Banos is encountering during the registration period. It is an NP-complete (non-deterministic polynomial time) problem which involves a set of constraints, mostly class size, student population, faculty availability and time constraints [3]. A good algorithm, which takes in consideration the constraints above, must be considered in order to generate the best fit timetable for all students. Currently, UPLB is using REGIST, an automation system that provides the initial school timetable for all registered students by predicting the demands for every subject that will be offered in the next semester [2]. Using the prediction data, the system will then “pre-enlist” the subjects on the students, therefore creating an initial schedule that the students themselves can modify during the registration period.

The system is designed to generate students' schedules in a priority queue based on their current academic standing and year in the curriculum. This implies that those with 'good' academic standing will be prioritized and is guaranteed to get all subjects that are stated in the curriculum for the current semester. However, most of the time, REGIST does not meet that requirement as it underestimates the demand that students should have, leaving most students with good standing with incomplete timetables. Also, the system has been inconsistent in providing the adequate schedule based on the curriculum as it does not consider the weight or priority of a specialization or major subject over a General Education (GE) subject. This poses a problem since most major subjects are offered seasonally, which hampers the student in following the curriculum of his degree.

In this study, we discuss the Automated Registration Guidance system (ARGS), or as a possible solution to address the concerns brought up by the inconsistencies of REGIST and to increase the number of students who will be given complete pre-enlisted schedules by the system. ARGS will take off from REGIST's initial features, and will improve its algorithm on pre-enlistment by using heuristic search algorithms. The system will analyze all curricula that is being used by different programs in the university; and generate N general schedules on all types of students per curricula using Backtracking algorithm. These general schedules will be distributed to all students until all slots have been exhausted. The GE subjects per student, will be generated using the Genetic Algorithm, which employs gene-based functions such as crossover and mutation - finds the best fit subject based on the student's generated major subject timetable and curriculum. By the completion of the study, the system generated will not only be helpful in determining the most adequate scheduling algorithm for the university, but also contribute to the whole registration process as an alternative tool in providing pre-enlistment services to the students.

# RELATED WORK

There have been several studies that tackled the issue of class timetabling. However, there has been few researches that is leaning to the usage of backtracking as a main algorithm for this problem. Zelenski pointed out backtracking lessens the risk of the algorithm to be an expensive solution since it does not consider solutions that will not lead to the goal [11]. Schaerf, on the other hand, tells that timetable scheduling algorithms are difficult that a solution leaning to Artificial Intelligence must be considered [8].

With this premise, there has been an abundant number of studies on the Genetic Algorithm as a viable timetable scheduling solution. Abramson &Abela represented a chromosome as a set of tuples or subjects, and then grouped together to form a timetable. [1] Sigl, Golub and Mornar represented a timetable as a 3D array, to generate schedules using Genetic Algorithm at a short time.[10] A further expansion in speed optimization of Genetic Algorithms were performed by Hermann and Lee by reducing the runtime of the algorithm by extending the heuristic function in a multiple-pass method [5]. Lastly, study by Cherreguine in 2008 tested the capability of Genetic Algorithm in a local sense by generating a constraint-free schedule at the Institute of Computer Science in the university[3].

Other studies such as Chu & Beasley noted Genetic Algorithm as a superior solution in the general assignment problem. [4] Reeves, on the other hand, applied GA on the A|B|P|Cmax sequencing problem for a flowshop system [6].

Studies on backtracking were further elaborated by Shmidt and Druffel in testing directed graph for isomorphism at O(n2) running time [9]. On the other hand, Sadeh, Sycara and Xiong defined backtracking as a viable solution for NP-Complete solutions. However, they pointed out that backtracking alone presents a “deterioration of the efficiency of the solution” by increasing the time constraints so a heuristic must be used to improve it [7].

There has been a sufficient study on the usability and effectiveness of both Genetic and Backtracking algorithms in terms of its usage as a scheduling algorithm not only on timetables, However, there has been little or no mention of a study that will combine the two solutions on a single system. The study aims to extract the enumerative features of backtracking, and the heuristic capability of Genetic Algorithm into a one single system, generating a complete and less constraints for all timetables needed in the system.

# METHODOLOGY

The study involves two algorithms that use different data structures in getting the solution. The Genetic Algorithm's basic data structure is the chromosome, which is basically a strand or a list of related data where genetic operations are conducted. Backtracking, on the other hand, operates on a tree-like structure in iterating the possible combinations of a schedule. However, they both operate on the same data set: the timetable. Therefore, a standard structure on the timetable system must be devised in order for the two algorithms to communicate with each other.

The timetable will consist of an array or a vector of linked lists, in which every element of the array represent a half an hour (30 min) time period. This is because the university has classes that begins or ends on a half-hour chunk. Each element, or chunk, is a linked list that accept a subject block. If a subject is 3 hours long, it will have 6 subject chunks to occupy in the system, if it is 1 hour, it will have 2 chunks, and so on.

*Recursive Backtracking Schedule Generation*

The goal of the recursive backtracking algorithm in the system is given a curriculum, and the semester and year level it will generate on, the system will generate schedules from the assumption of a “model student” (that is, a student with no failing grade, on time with the curriculum). This assumption will help the system to exhaust all possible combinations of subjects that the “model student” can take. The generated schedules, on the other hand, will take form as schedule buckets, wherein students will be granted a slot by filling in one of these buckets. If a certain student is not permitted to take the subject, it will not be enlisted and will be treated as a free slot. The process will repeat until all buckets are filled, or all students have been assigned to a bucket. The assignment will still make use of the priority queue scheme, similar to REGIST's priority system. The algorithm will repeat until all curricula and year levels in the university have been generated.

*Genetic Algorithm*

The generated schedules for all students from the backtracking algorithm will be completed using the Genetic Algorithm method. Since we have already pre-enlisted the major subjects on every timetable, we will modify the chromosome generation so that the major subjects will not be altered by the GA operators. Given a student's recommended General Education (GE) subjects, PE Subjects, and NSTP Subjects, and Major Subjects that is not part of the subjects enlisted by the Backtracking algorithm (irregular Major Subjects), the system will generate a pool of subjects containing all schedules of lectures and laboratories. A linked list will serve as a mini-chromosome that will randomly pick from the pool of subjects stated above. This mini chromosome will be incorporated on the main chromosome (the timetable) by “trying to enlist” the subjects contained in the mini chromosome.

The fit of the mini chromosome to the main chromosome will be determined by the fitness function. The function will be of a points system from 0 to 1, which is determined by a heuristic (which checks whether there are overlapping classes, faculty availability, et al).If the fitness of the chromosome does not reach to a threshold (which can be variated) the system will employ either crossover (classes from two mini-chromosomes and cross the subjects to form another mini chromosome) or mutation (a random subject from the mini checklist will be replaced by another random subject in the pool). This will repeat until the threshold number for the fitness value has been achieved.

# EVALUATION

The system will be evaluated by testing the system on a mock enrollment process that will be divided into two parts: First, enrollment data which include a set of curriculum, a set of active undergraduate students, and a set of currently offered subjects and their sections will be used as the primary input. These inputs will be fed to both REGIST and ARGS and both system will run its pre-enlistment methods in parallel with each other.

The effectivity of ARGS will be measured by comparing its results with REGIST. The data that will be measured are:

1. The number of unique schedules generated given the schedule and class size constraints.

2. The percentage of students with 'Good' standing that received a complete number of units

4. The distribution of students in terms of the number of units received (100%, 80%, 50%, 20%, 0%)

5. The number of free slots or unallocated slots after the pre-enlistment process.

6. The total number of time elapsed in generating the schedules of all students.

The second part of the evaluation process will involve the testing of the integrity and quality of schedules generated per student. To do this, a random sample of 100 students from different degrees and year levels will be selected. They will be clustered according to their classification (Freshman, Sophomore, Junior, Senior) to view the initial schedule generated by the system being developed. A satisfaction rating from 1 to 100 will be asked to the respondents, their suggestions on what subject the system should generate if the system chose a wrong subject or did not enlist a subject, as well as their suggestions in improving the generated initial schedule. The performance of the system will be evaluated by getting the average satisfaction rating from all the respondents, as well as the feedback received from them.

# TIMELINE

* **DATA GATHERING:** Necessary data will be gathered such as class schedules in a semester, curriculum of degree programs offered, list of enlisted students and previous data generated from REGIST.
* **SYSTEM IMPLEMENTATION:** The system will be implemented using Backtracking and Genetic Algorithm and the data gathered. The coding process will be divided into three stages: First, The data storage schemes for the schedules or timetable, then the implementations for the Backtracking and Genetic Algorithm. The last scheme to be implemented is the connection between the two algorithms to generate the schedule and maintain the class list slots.
* **SYSTEM TESTING AND EVALUATION:** After the implementation phase, the system will be prepared for user testing and evaluation.The data gathered from the system will be compared head-to-head with REGIST, then a random sample of students per classification (Freshman, Sophomore, Junior, Senior) will participate in the evaluation of the system. A satisfaction survey will be done during the process.The result of the evaluation will be analyzed in order to assess if the solution is feasible.
* A final evaluation will be performed to sum up the results given by the system on whether it solved the problem on pre-enlistment. Also, an assessment whether the system is ready for live deployment.

# REFERENCES

1. Abramson, D., & Abela, J. (1992). A Parallel Genetic Algorithm for Solving the School Timetabling Problem. *15 Australian Computer Science Converence, Hobart* , 1-11.

2. Carino, R. L. (n.d.). *Heeds*. Retrieved November 5, 2015, from Github: https://github.com/rlcarino/heeds

3. Cherreguine, J. L., & Albacea, E. A. (2008). *ICS Timetable Automation Using Genetic Algorithm.* Los Banos, Laguna: Institute of Computer Science, University of the Philippines.

4. Chu, P. C., & Beasley, J. E. (1997). A Genetic Algorithm for the Generalized Assignment Problem. *Computers Ops Res. Vol. 24, No. 1* , 17-23.

5. Herrmann, J. W., & Lee, C.-Y. (1995). Solving a Class Scheduling Problem with a Genetic Algorithm. *ORSA Journal on Computing Vol.7, No. 4* , 443-452.

6. Reeves, C. R. (1994). A Genetic Algorithm for Flowshop Sequencing. *Computers Ops Res. Vol. 22, No 1* , 5-13.

7. Sadeh, N., Sycara, K., & Xiong, Y. (1992). *Backtracking Techniques for Hard Scheduling Problems.* Pittsburgh, Pennsylvania: Carnegie Mellon University.

8. Scheaerf, A. (1999). A Survey of Automated Timetabling. *Artificial Intelligence Review 13* , 87-127.

9. Schmidt, D. C., & Druffel, L. E. (1976). A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices. *Journal of the Association for Computing Machinery, Vol 23 No. 3* , 433-445.

10. Sigl, B., Golub, M., & Mornar, V. (2003). *Solving Timetable Scheduling Problem by Using Genetic Algorithm.* Unska 3, 10000 Zagreb, Croatia: Faculty of Electrical Engineering and Computing, University of Zagreb.

11. Zelenski, J. (2008, February 1). Exhaustive Recursion and Backtracking.