Fayoum University Faculty of Engineering, 2023, Vol: 6(1), 39-46
https://fuje.journals.ekb.eg/
ISSN Online: 2537-0634
ISSN Print: 2537-0626

University Examination Timetable Scheduling Using Constructive Heuristic Compared to Genetic Algorithm

Dina A. Salem[1,*]

[1]Computer Engineering Department, Faculty of Engineering, MUST University, Giza 12566, Egypt

*Corresponding author: Dina A. Salem (dina.almahdy@must.edu.eg).

How to cite this paper: Salem, D.A. (2023). University Examination Timetable Scheduling Using Constructive Heuristic Compared to Genetic Algorithm. *Fayoum University Journal of Engineering, 6,* 39-46
https://dx.doi.org/10.21608/FUJE.2022.157195.1018

Abstract

Examination Timetable Scheduling can be defined as assigning a set of exams into a limited number of days and periods, subject to a set of constraints, some of which contradict each other. An issue that makes timetabling a challenging task most universities have to solve every semester. It usually takes days or even weeks to find a solution that does not even "entirely" satisfy the universities' constraints. Automating this task is really challenging, since timetabling problem is a Nondeterministic Polynomial time (NP) complete problem. The most known characteristic of NP-complete problems is that no fast solution to them is known. That is, the time required to solve the problem using any currently known algorithm increases exponentially as the size of the problem grows. NP-complete problems have tremendous solution space, so brute-force methods cannot be used . This research builds and compares the performance of two proposed solutions; Genetic Algorithm and Constructive Heuristic models when both used to solve a real-world examination timetabling problem at Misr University for Science and Technology (MUST). This research shows how the Constructive Heuristic proposed outperformed the Genetic Algorithm in creating optimized schedules which solves all the students' conflicts. An easy to use, GUI desktop application was created to be used by all universities' members, to automate their timetabling tasks, the program creates many feasible schedules, meeting all requirements of MUST examination timetables, while reducing the pressure on the students, in seconds, which will result in saving a lot of time and effort for the universities' members.

Keywords

Constructive Heuristics; Examination Timetabling; Genetic Algorithm; Scheduling;

1. Introduction

Generally, Scheduling is the process of planning and fitting specific tasks that must take place in available and constrained time slots (Burke, 1996). Timetabling problem is an excessive restricted scheduling problem which is applied to many fields with the goal of assigning available (often limited) activities' resources to fulfill essential objectives. These fields include but not limited to military, governmental institutions, transportation, industry, communication, aviation, and education (Lewis, 2008; Qu et al., 2009b; Aldeeb at al., 2019).

Educational institutes consume a lot of time to prepare their different timetables including High School Timetabling, University Course Timetabling and Examination Timetabling (Ceschia et al., 2022). For decades, examination timetabling has become one of the most studied domains in Artificial Intelligence. This is due to its importance in many academic institutions worldwide. Examination Timetabling can be defined as assigning a set of exams into a limited number of days and periods, subject to a set of constraints. (Emmanuel et al., 2022)

The greater the number of constraints in the problem, the more difficult the exam timetabling problem becomes. In addition, some constraints contradict each other, which makes timetabling a challenging task, and most universities have to solve it "manually" every semester. Besides, there is a great number of different constraints that make exam-timetabling problems different from one institution to the other. It usually takes days or even weeks to find a solution that does not even entirely satisfy the universities' constraints. Timetabling constraints are of two types, hard constraints, and soft constraints. Hard constraints must be satisfied in a strict manner where timetabling solutions that cause hard constraint violations (infeasible solutions) are typically considered to be invalid solutions. Soft constraints express a preference among feasible solutions and are often used to measure the quality of the schedule. A feasible solution is a one that satisfies all hard constraints (Rossi-Doria et al., 2003).

If a schedule is feasible, then the level of satisfaction of the soft constraints can be a measure of the quality of a timetable. Scheduling problems are NP-complete problems; thus, the solution space is humongous. For the examination timetabling problem, the size of the solution space can be calculated by knowing the number of start times (periods) and the number of exams. E.g.: if there are P periods and E exams, then there are P^E possible schedules. An algorithm can be built that can create all the possible schedules and creating them will not require a lot of time, but the problem is, how to determine a schedules' feasibility and quality. Evaluating the schedule is the process that actually requires most of the computation time. As a result, brute forcing is not an option, so, this research proposes two different methods: an optimization Algorithm (The Genetic Algorithm - GA) and a Constructive Heuristic. Both methods produced feasible schedules with the implemented constructive heuristic approach was able to satisfy all the soft constraints in less time.

The rest of this article is organized as follows; section 2 summarizes the main work done on examinations timetabling problem and highlights the main contribution of this research. section 3 describes the two proposed methods to solve the problem at hand. Section 4 lists and analyzes the results. Section 5 concludes the research.

2. Related Work

Examinations timetabling is one main topic that attracts researchers' attention decades ago. Changing environment and constraints from time to time makes scheduling with all its applications a hot subject that is in deep need of continuous modifications.

In 1996, research was conducted in an attempt to create a general algorithm that is able to be generalized for many institutions. Despite being able to produce flexible solutions, the mentioned models suffer from running time and intensive parameters that need tuning (Thompson & Dowsland, 1996). Further work is done in 2007 that uses ant colony algorithms on the online Toronto benchmark dataset. Running time and performance of used local search was reported as models' drawbacks (Eley, 2007).

An optimization algorithm called honey-bee mating was implemented to solve the examination timetabling problem. It achieved promising results on two datasets that are widely studied but not being tried on a real-life dataset (Nasser et al., 2016). In 2022, a survey was done to explain the main pros and cons of several timetabling approaches. The survey concludes that there is still a need to enhance the algorithms used to produce timetables (Ceschia et. Al. 2022).

Out of previously explained, this research was carried out to solve the examination timetabling problem using MUST dataset as a real-life example. The proposed model succeeded to produce feasible schedules in a very adequate time satisfying all hard and soft constraints. The main contribution of this research is that the created model is flexible and can be adapted to many scheduling applications. However, it is very well tailored to an Egyptian university (as a practical example) covering all its limited resources and resulting in an excellent performance.

3. Methodology

This research main task is to automate the examination timetabling process by creating an easy-to-use desktop application. This application aims to create a schedule that satisfies all the hard constraints while satisfying as many soft constraints as possible. Hard constraints are considered as: a student shall not have two exams at the same time, a student shall not have three exams on the same day, and students in special courses (courses that require more examination time than the other courses) shall not have a second exam on the same day as a special course. While the soft constraints are as follows: reduce the pressure on students by spreading their exams, schedule popular courses first, and maximize the number of students/exams on Fridays. The main goal is to create a schedule that does not only satisfy hard constraints but also the soft constraints, as many as possible. To create such a program and automate the scheduling process, some of the faculties' data is used as input to the program. The dataset contains twelve inputs that are mentioned in **table 1**.

After acquiring all the inputs needed, the user can then start the scheduling process and the program will report whenever a feasible schedule is created and its quality, the user later can save any of the feasible schedules created choosing the best one that fits their needs. Then the program will write the schedule to the disk in .xlsx format with a hardcoded template.

Table 1. List of all the inputs applied to the proposed model

	List of inputs
1.	First day of the exams
2.	Last day of the exams
3.	Excluded days from the schedule (vacations or holidays)
4.	How many periods per day
5.	Specify each period start and end time
6.	Exclude specific periods from specific days e.g., Friday usually got fewer periods than other days
7.	Specify the maximum number of exams per period
8.	Specify the maximum number of students per period
9.	All the courses that should be on the schedule
10.	All the students registered in the semester and the courses that they are registered in
11.	Specify the special courses (courses that require more examination time than most of the courses)
12.	Choose any or both constraints that should be applied on periods (Limit the number of students per period and limit the number of courses per period).

3.1. Evaluation Function

The most computationally expensive function in the whole program is the evaluation function, sometimes called the fitness function, its role is first to decide whether a schedule is feasible or not, then, evaluates it to determine its quality as shown in **figure 1** (quality of the schedule is checked only if it is feasible). The evaluation function is divided into two modules: Feasibility check and Evaluation of pressure on students.

After calling the function and passing the schedule to it, it first passes the schedule to the feasibility check method. The feasibility check method returns true if and only if the schedule contains all of the required courses, no student has two exams at the same time, no student has more than two exams in the same day and no student has two exams in the same day if one of the exams is of a special course. If it returns true, it will then be evaluated and given a pressure value, which is an integer, defining how many students have consecutive exams on the same day, the lower the number is, the better the schedule. If the schedule is not feasible it will not be evaluated, so it will be discarded.

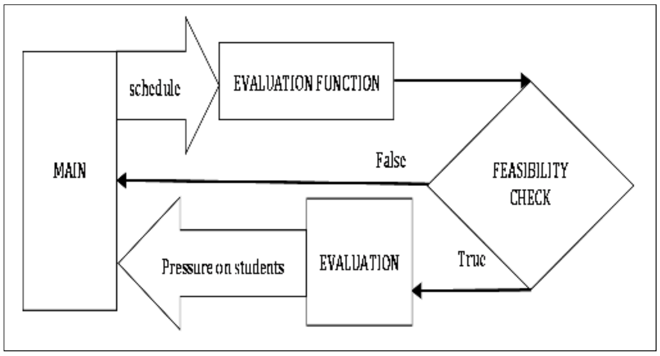

Figure 1. Evaluation function

Why to discard unfeasible schedules? There is a chance that a schedule might not be created at all! True, but actually the constructive heuristic, was able to create a lot of feasible schedules (tested on inputs of four different semesters). The program never gets stuck! So, a new challenge is to choose the best schedule, from the feasible ones. If the schedule is feasible (feasibility check returns true), the schedule is then evaluated by the evaluation function, which evaluates it based on the total number of students that have two exams at the same time and returns that number to the main function.

3.2. Genetic Algorithm

Genetic Algorithm (GA) is a metaheuristic approach that is known to have good solution in many optimization problems (Salem et al., 2012). Basic Steps of Genetic Algorithm are shown in **figure 2.** One of the most important decisions to make while implementing a genetic algorithm is deciding the representation that will be used to represent the solutions.

It has been observed that improper representation can lead to the poor performance of the GA. Therefore, choosing a proper representation, having a proper definition of the mappings between the phenotype and genotype spaces is essential for the success of a GA. Out of caring about the order of the Alleles, so the representation chosen is that of the Traveling Salesman Problem (TSP) (Applegate et al., 2007) since the permutation representation is the most suited one.

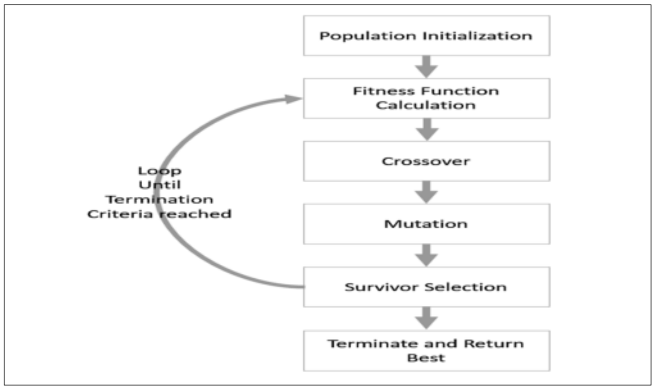
Population Initialization
Fitness Function Calculation
Crossover
Mutation
Survivor Selection
Terminate and Return Best
Loop Until Termination Criteria reached

Figure 2. Genetic algorithm basic structure

The schedule is an array of courses, where every Gene (Allele's position) represents the date and time where a course is going to be scheduled, and the Allele itself is an integer representing the course. All courses' names are encoded (mapped) to integers for this representation. An instance is clarified in **Figure 3** that represents a schedule for two days. Day 1 starts from index [0] to index [4], while day 2 starts from index [5] to index [9]. Each Gene (index) in a day indicates a specific period and time, in which that course will take place. The Allele (integer value) itself is the course name.

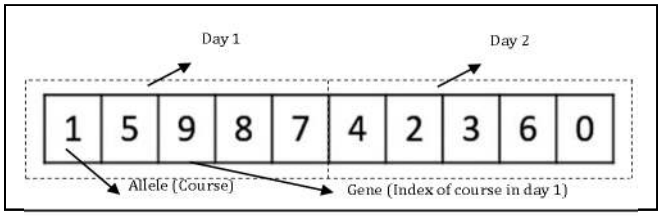

Figure 3. Genetic algorithm representation of employed model.

The flow chart in **figure 4** briefly explains the steps of GA applied to reach the best reliable timetable. The selection operator "Survival of the Fittest" is used to select the best candidates from the current population and then all the selected candidates are added to the mutation pool for mating. Two candidates are then selected to mate which are called “Parents”. Offspring are created by applying either crossover (combining the vector entries of a pair of parents – Davis’ order crossover operator “OX1”) or mutation (making random changes to the parents – Swap Mutation) or both, to the parents. The new population now, is, all the newly created off-springs, which are probably more fit than their parents. The process is repeated until a termination condition is reached.

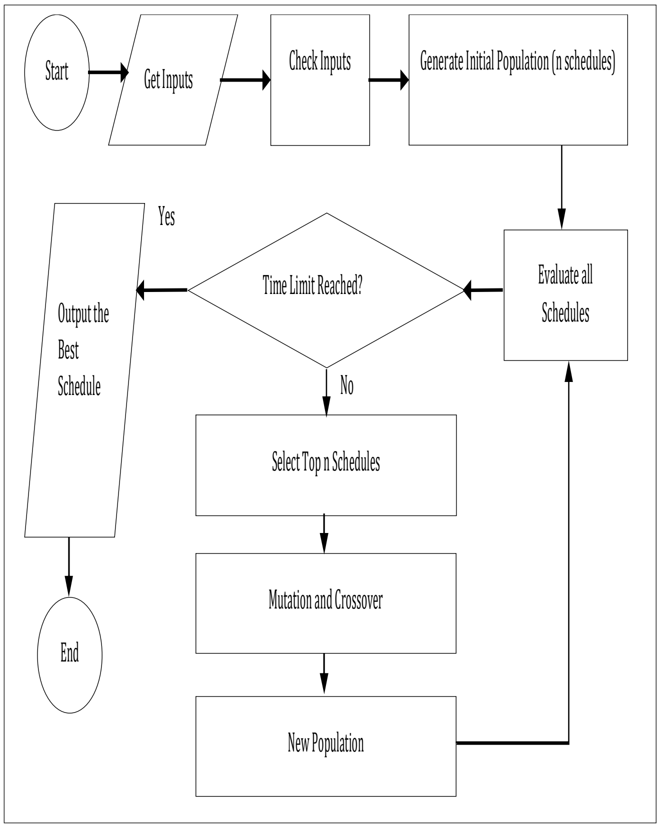

Figure 4. Applied genetic algorithm flowchart.

Given the number of days, periods and courses, the initial population is created, which is several randomly created schedules. These randomly created schedules are created by first initializing an empty array with the size of the schedule, then randomly choose courses and add them to random locations to fill up the array (schedule). All of the initial population schedules are then evaluated. If the result does not have a good enough schedule, proceed with the GA.

3.3. Applied Constructive Heuristic

A constructive heuristic is a type of heuristic method which starts with an empty solution and repeatedly extends the current solution until a complete solution is obtained. Examples of some constructive heuristics developed for famous problems are flow shop scheduling (Koulamas, 1998), vehicle routing problem (Petch, 2003), open shop problem (Bräsel, Tautenhahn, & Werner, 1993).

This research uses constructive heuristic to solve the examination timetabling problem, that works exactly as a human who will try to create the timetable manually. It starts by taking the inputs defined earlier at the beginning of this section. And mostly depends on the order of the list of courses. Because it fills the schedule starting from the first, based on the hard and soft constraints that are hard-coded in the program, once it finds a suitable slot for that course it will place it in that slot, then moves on to the next course from the courses list. It keeps filling slots by courses until it reaches one of two end points, either all courses have been scheduled or reach what is called a dead-end. Dead-ends usually occur when the program succeeds to schedule majority of the courses but fails to schedule two to ten courses. This failure usually takes place when the available empty slots are not suitable for the remaining courses and do not meet the required constraints. The steps applied to the constructive heuristic approach to generate the required examination timetable are illustrated in **figure 5**.

Inputs to the model are list of courses, maximum exams per day for a student and a preliminary schedule (if available). It starts with picking the first course from the list of courses. Then it loops overall schedule slots until a feasible slot is found and place the course in that slot. It keeps repeating to pick the next course and find the appropriate slot. If all courses are scheduled successfully, it returns the schedule. If some courses are not scheduled, it returns the schedule and the list of courses that were not scheduled.

After calling the heuristic function, if it fails, a second call can take place, but with the list of courses that are returned, which are the courses that were not successfully scheduled. However, the program changes the number of exams per day which allows for a very good chance that these courses will then be scheduled. So, as a first try the program schedules all courses by setting the maximum number of exams to one. Then, call the heuristic function. If it fails, the program calls it again, with the list of un-scheduled courses and the created schedule that is missing those courses, then increment the maximum number of exams to two. This proposed approach eventually succeeds to reach very good solutions.

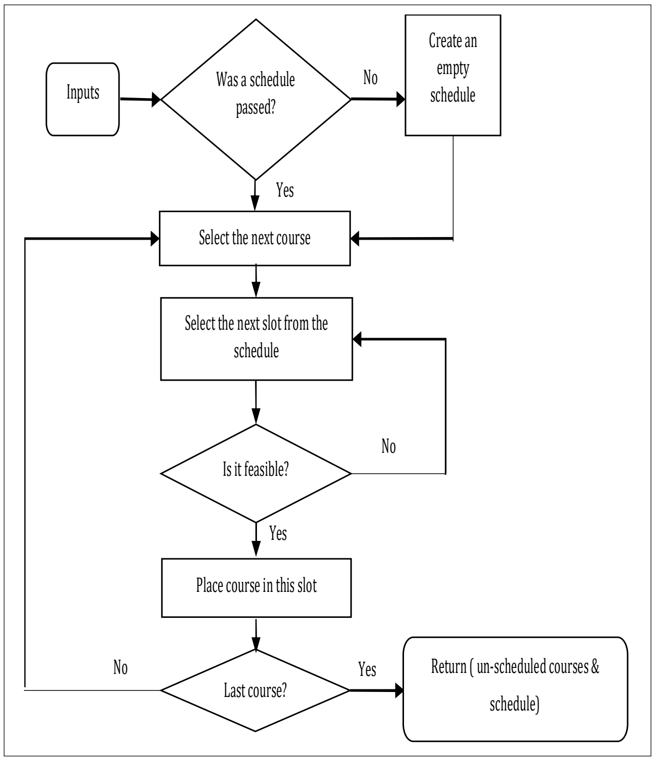

Figure 5. A Constructive Heuristic Flow Chart.

4. Results

To evaluate the two proposed models, a real dataset is used from Faculty of Engineering-Misr University for science and Technology (MUST). The faculty operating in a credit hours system, offers three semesters per year: Fall, Spring and Summer. About 220 courses are available per semester, and the final exams usually takes around 18 days (including holidays), with 3 periods per day. So, the number of possible schedules is 220^{54} or 3×10^{127}. An application is successfully created that can save about four weeks of time for faculty members who are responsible for producing the examination timetable schedule. In addition to helping the students perform better in their exams by reducing the number of exams they have in each day.

4.1. Genetic Algorithm Results

The proposed GA with the applied parameters' optimization was able to produce a feasible schedule in an average time of 4 minutes. However, the output schedule despite being feasible satisfied the hard constraints only, but not the soft constraints. **Figure 6.a and 6.b** shows the results of the applied GA model. **Figure 6.a** clarifies the average time when applying the final GA model to the faculty datasets of different semesters. **Figure 6.b** assumes that the accepted fitness score for a feasible schedule is 10 and accordingly concludes that GA wastes a lot of time trying to just create a feasible schedule.

4.2. Constructive Heuristic Results

At first, Constructive heuristic (CH) was proposed to be used to create the initial population for the GA to reduce time. It creates a population of feasible schedules and pass them to the GA as the initial population so that the GA can optimize them and produce better results.

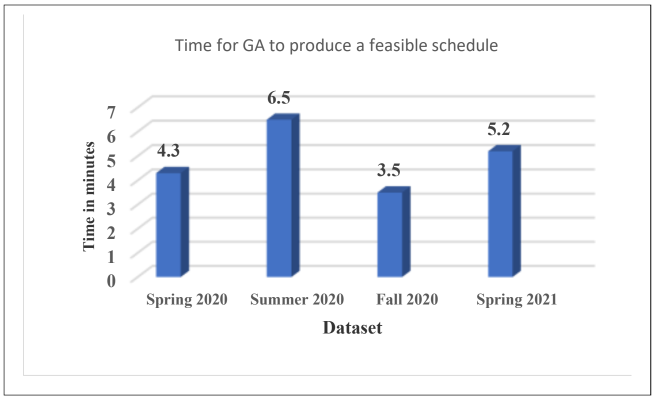

Figure 6.a. Results of Applying the Final GA Model to Different Datasets.

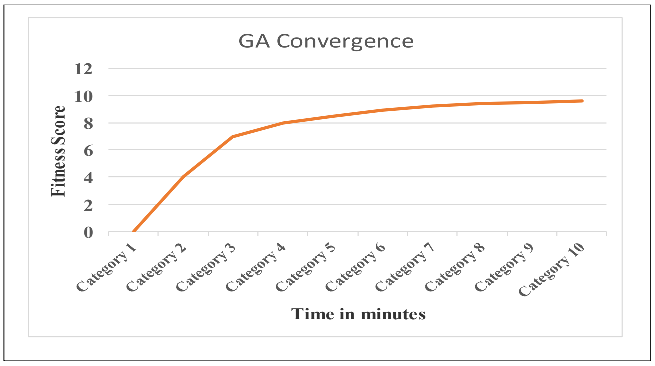

Figure 6.b. Time taken by GA to reach a fitness score equals 10

But after several trials optimizing the constructive heuristic and rewriting its code trying to make it produce the best results, the constructive heuristic was able to produce feasible and optimized schedules on its own, in a matter of seconds with the results recorded in **figure 7.**

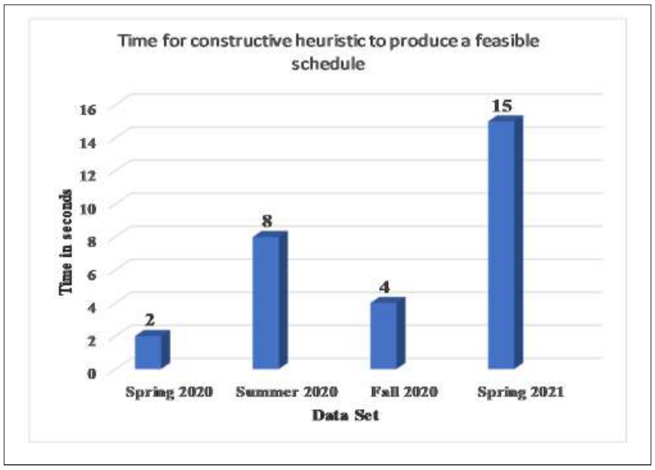

Figure 7. Results of applying CH model to different datasets.

4.3. GA versus CH

To fully evaluate the performance of the two proposed models on the mentioned dataset, a separate run is carried out to record the number of students with two exams at the same day. In this run time parameter is set to 1 minute and the same four datasets are used, and results are recorded in **Figure 8.**

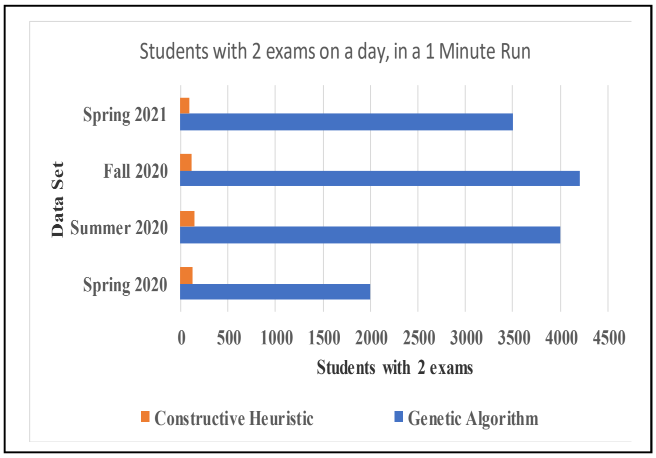

Figure 8. Results chart to compare GA and CH.

5. Conclusion

This research proposed two different methods to solve the examination timetabling problem for MUST (as an example), genetic algorithms and heuristic approach. Both techniques are implemented using python programming language and a GUI scheduler is created based on the MVC design pattern to be used to automate the process of scheduling.

The implemented approaches were tested on 4 different datasets. This showed that the constructive heuristic is way faster and produces much better schedules than the GA., and it worked flawlessly.

As the proposed CH model was proven to work properly on MUST engineering datasets, it can be further applied to other faculties of the same university as constraints are very similar. Although CH approaches are tailored to specific problems, but they are generally characterized by being flexible which makes the model easily transferred to other universities with slight modifications. The model can also be generalized to many examinations timetabling including schools and online exams.

References

Aldeeb, A. B., Al-Betar, M. A., Abdelmajeed, A. O., Younes, M. J., AlKenani, M., Alomoush, W., Alissa, K. A., & Alqahtani, M. A. (2019). A Comprehensive Review of Uncapacitated University Examination Timetabling Problem. *International Journal of*

Applied Engineering Research. Vol. 14 (pp. 4524-4547). Research India Publications.

Applegate, D. L., Bixby, R. E., Chvatal, V. & Cook, W. J., (2007). The Traveling Salesman Problem: A Computational Study. Princeton University Press, United States.

Bräsel, H., Tautenhahn, T. & Werner, F. (1993). Constructive heuristic algorithms for the open shop problem. *Computing*, 51(pp. 95-110).

Burke, E. K., Elliman, D. G., Ford, P. H. & Weare, R. F. (1996). Examination timetabling in British universities - A survey. In: Burke, E.K., Ross, P. (eds.) *Practice and Theory of Automated Timetabling. LNCS. Vol. 1153* (pp. 76–92). Springer, Heidelberg

Ceschia, S., Gaspero, L. D. , & Scherf, a., (2022). Educational timetabling: Problems, Benchmarks, and State-of-the-art Results. *European Journal of Operational Research. In press.* https://doi.org/10.1016/j.ejor.2022.07.011.

Eley, M. (2007). Ant Algorithms for the Exam Timetabling Problem. In: Burke, E.K., Rudová, H. (eds) Practice and Theory of Automated Timetabling VI. PATAT 2006. Lecture Notes in Computer Science, vol 3867. Springer, Berlin, Heidelberg.

Emmanuel, M. C., Chidimma, N. C., Sunday, B. T. & Ejike, C. O. (2022). Re-Engineering of Examination Timetabling Generation and Invigilation Scheduling System. *International Journal of Innovative Science and Research Technology Vol. 7* (pp. 662–634). Springer, Heidelberg

Koulamas, C. (1998). A new constructive heuristic for the flowshop scheduling problem. *European Journal of Operational Research, Vol. 105(1),* (pp. 66 – 71)

Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. *OR spectrum, Vol. 30 (1)*, (pp. 167–190).

Petch, R. J. (2003). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics, Vol. 133(1-3),* pp. (69 – 92).

Qu, R., Burke, E. K., McCollum, B., Merlot L. T. & Lee, S. Y. (2009b). A survey of search methodologies and automated system development for examination timetabling. *Journal of scheduling, Vol. 12 (1),* (pp. 55–89).

Rossi-Doria, O. et al. (2003). A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. In: Burke, E., De Causmaecker, P. (eds) Practice and Theory of Automated Timetabling IV. PATAT 2002. *Lecture Notes in Computer Science, Vol 2740*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45157-0_22

Sabar, N.R., Ayob, M., Kendall, G., & Qu, R. (2012). A honey-bee mating optimization algorithm for educational timetabling problems. Eur. J. Oper. Res., 216, 533-543.

Salem, D. A., Abulseoud, R. A., & Ali, H. A. (2012). K5. merging genetic algorithm with different classifiers for cancer classification using microarrays. 2012 29th National Radio Science Conference (pp. 659-666). doi: 10.1109/NRSC.2012.6208579.

Thompson, J., Dowsland, K.A. (1996). General cooling schedules for a simulated annealing based timetabling system. In: Burke, E., Ross, P. (eds) Practice and Theory of Automated Timetabling. PATAT 1995. Lecture Notes in Computer Science, vol 1153. Springer, Berlin, Heidelberg.