Open Journal of Science and Technology
Journal Homepage: http://readersinsight.net/OJST

Research Article

Development of a generic timetabling and scheduling system based on the combinations of Meta-Heuristics algorithm

Muhammad Raheel[1]*, Muhammad Adeel[2], Muhammad Nabeel[3], Muhammad Sajeel[4]

[1]Department of Data Science, Universität Bielefeld, Bielefeld, North Rhine-Westphalia, Germany
[2]Department of Human Computer Interaction, Universität Siegen, Siegen, North Rhine-Westphalia, Germany
[3]Department of Information Technology and Engineering, Frankfurt University of Applied Sciences, Frankfurt am Main, Hessen, Germany
[4]Department of Computer Science, Sir Syed University of Engineering & Technology, Karachi, Pakistan

*Corresponding Author email: muhammadraheel_95@yahoo.com

Submitted: 15 September 2021 *Revised: 20 December 2021* *Accepted: 27 December 2021*

ABSTRACT

This research paper examines the usage of combining the meta-heuristic approaches for solving critical real-world optimization problems due to the hard and soft constraints. For the last few decades, developing an optimal solution for these problems were considered as a difficult task, and manual timetabling required an excessive amount of effort and time to deliver the quality results. This paper's objective is to address several approaches and develop a generic solution for solving a wide range of timetabling and scheduling problems. Many techniques and benchmark instances have been reviewed and presented to find a feasible solution in the literature section. This paper also investigates the performance based on the combination of meta-heuristic approaches to solve the problem and introduce the algorithm to allocate time slots to several resources according to their constraints satisfaction and violations. However, this algorithm may consider as a comprehensive benchmark instance and significant improvement for future researchers. Furthermore, the computational and experimental results of the algorithm determine the feasibility, effectiveness, robustness, and optimality of the proposed approach.

Keywords: *Timetabling & Scheduling; Meta-Heuristic; Optimization; Performance; Feasibility*

1. INTRODUCTION

The main topic of this dissertation is to understand the limitations and challenges of timetabling and scheduling problems (TSP) and to develop a generic optimal solution.

1.1. MOTIVATION AND OVERVIEW

On various real-world timetabling and scheduling problems (TSP) faced by every individual and organization on an everyday basis which is described below in the problem statement section. However, the tradition based timetabling approaches and methods intimately related by hand; therefore, these methods are not sufficient to satisfy the constraints and requirements. Also, the problems are hard to develop by using operation research

https://doi.org/10.31580/ojst.v4i4.2508
© 2021 The Authors. Published by Readers Insight Publisher. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/)

approaches and knowledge-based approaches, because these approaches are not fast enough and take more time in computation and they are also not flexible for different problems (1).

In the past several decades, many research papers have been published on the various timetabling and scheduling algorithms that have been proposed by researchers such as constraint-based methods (2), meta-heuristic methods (e.g., tabu search (TS)(3), simulated annealing (SA) (4) etc. In general, a population-based algorithm based on Genetic Algorithm (GA) (5) is more powerful and efficient in solving timetabling problems because of the searching capability within the whole space, but the drawback of population-based algorithm is quiring more time that can be overcome by implementing ranking techniques and Statistical Algorithm (6). Besides, by introducing a statistical algorithm, guided search and ranking techniques to improve the performance of the genetic algorithm for timetabling problems, and there is a possibility of future aspects and further advancements after the completion of this paper.

1.2. Scientific Contribution & Goals

This paper examines the effectiveness and efficiency of the combinations of metaheuristic algorithms based on experimental analysis for the timetabling and scheduling problems.

The goal of the paper is to build an automatic, effective and robust timetabling and scheduling system, which is used to monitor and allocate all the possible free slots to the resources. Some obvious research questions to accomplish the goals are as follows:

- How to select the desirable approach and algorithm to resolve the timetabling and scheduling problems?
- How to make a generic and efficient solution by using predefined algorithms?
- How to estimate the computational time and memory consumption?
- How to perform an algorithm to allocate all potential resources?

The overall research objective of the paper is to develop and implement the combinations of meta-heuristic algorithms to construct an optimal solution for TSP problems. Numerous points are defined as follows to achieve the aims, which are:

- To recognize the limitations and challenges of the following TSP and examine the currently applicable algorithms from researchers.
- Investigating the capability and competence of the classical (manual) solutions for the timetabling and scheduling problems.
- To implement the existing appropriate algorithms to obtain a generic solution, using the PHP, MySQL and JavaScript programming languages.
- To review search and heuristic techniques to develop a solution that is healthy, general and optimal for all real-world problems.

Furthermore, the other aim of the paper is to study the search approach and metaheuristic algorithms, and how these approaches provide an adequate and optimal solution for the timetabling and scheduling problems.

2. LITERATURE REVIEW

The timetabling and scheduling problems arise in various areas, such as transportation & logistics, training & education sector, health care organizations, etc. In contrast, some of the basic terms which are used in timetabling and scheduling problems are given below in Table 1.

Table 1. Basic terms used in timetabling and scheduling problems (7)

Terms	Description
Activity	An activity or event that needs to be scheduled.
Time Slot	A breach of time.
Resource	Availability of materials, assets etc.
Constraint	Limitations in organization or scheduling of an event.
Individual	A person who is willing to attend the event.
Conflict	A clash between two or more events.

2.1. TIMETABLING PROBLEMS

A timetabling problem is a specific type of problem in which the allocation of resources to time slots concerning differing constraints. The approaches for solving a substantial timetabling problem considering various constraints as a difficult job to be done. Due to the fact, a timetabling of events or activities such as subjects, exams or classes, etc., have to be organized in the form of time slots that satisfies all the limitations and restrictions. Let's consider, l as lectures and t as time slots, and according to the constraints of the problem is the potential timetable which means the most effective possible solution for a particular situation or resources. Traditional approaches are responsible for finding the most straightforward and shortest time slots that satisfy all the constraints, commonly used graph coloring algorithm.

Many researchers since the beginning of the 1960s are working on solving the timetabling problems, and till now, new researches are still emerging into this area (8). The most familiar example of the timetabling problem is education timetabling problems, which can be categorized in three different types based on their various constraints for each problem such as, school timetabling, course timetabling and exam timetabling. Nonetheless, the foundation of education timetabling is to assign the resources to time slots by reducing the constraint breaching, in spite of that, these problems are different from each other due to their constraints and existence.

2.2. SCHEDULING PROBLEMS

According to (9), a scheduling problem is a problem in which time must be allocated to resources to fulfill the task. Moreover, a nurse scheduling problem (NSP) which is also known as nurse rostering problem (NRP), is a problem of assigning the shifts to the nurses in a most pleasant way that must satisfy the set of soft and hard constraints for the quality of a solution (10). The assignment of nurses to shifts process is time-consuming, and the NSP is NP-hard.

2.2.1. Classical Solutions

A lot of time and effort requires to solve the timetabling and scheduling problems by using classical methods. The world fastest and powerful computer also needs a few decades to find all the potential combinations of the timetabling and scheduling problems solution with certain constraints. Due to the fact, researchers are searching for improved and optimal solutions methods, and mathematicians introduce the mathematical techniques to discard some of the potential combinations which cannot be a good fit as a solution for the particular problem. This research and implementations diminish the time consumption of finding the solution but not enough, which considered as an optimal solution. Therefore, to see the most elegant solution that takes inferior time and efforts, the heuristics approaches are taken into consideration.

2.2.2. Meta-heuristics

In mathematical optimization and computer science, a meta-heuristics approach is applicable for massive real-world problems. It is a general purpose approach to problem-solving within a search space containing the best, optimal and logical solutions, and this method can be applied in various optimization problems along with some modifications based on the specific question.

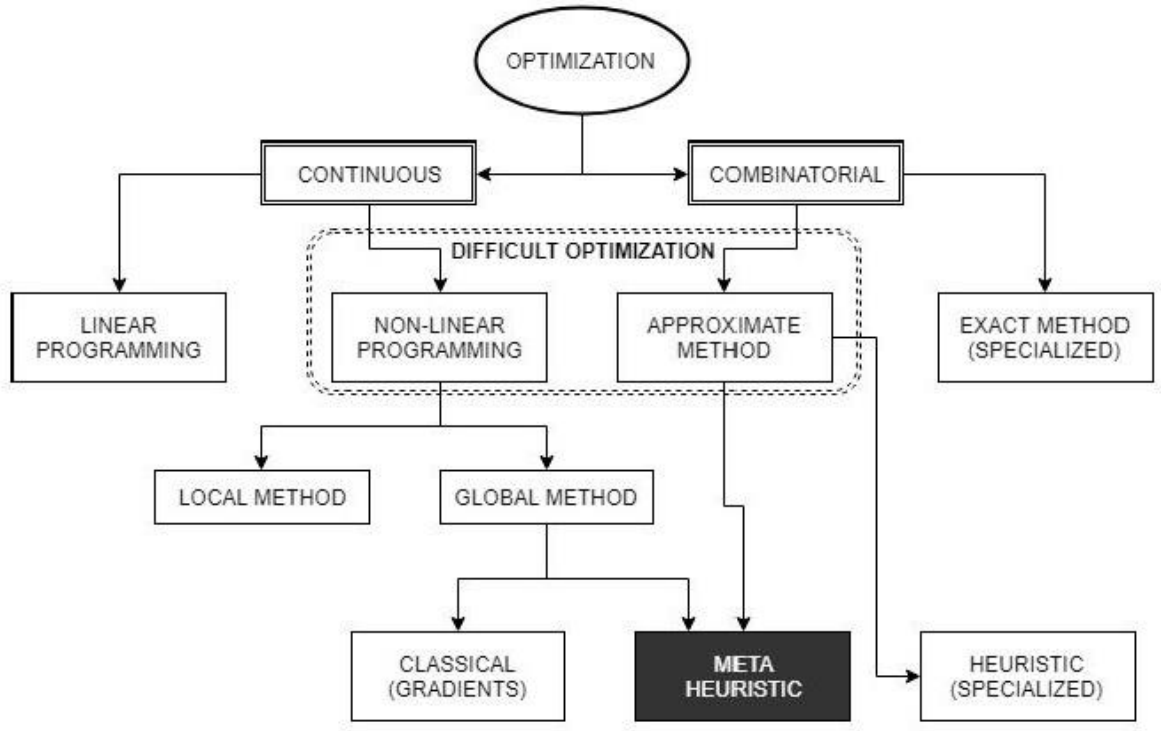


Fig. 1. Meta-heuristics underneath optimization classification (11)

Nowadays, the meta-heuristics methods occupied most of the engineering sectors, and the most known algorithms of meta-heuristics such as simulated annealing, ant colony optimization, tabu search, genetic algorithm, etc. According to (12), the term "meta-heuristic" was originally defined by (13).

"An iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search spaces using learning strategies to structure information in order to find efficiently near-optimal solutions."

2.2.3. Tabu Search

The basic concept of Tabu Search (TS) is used to solve the optimization problems.

"Tabu search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality." (14)

Tabu Search is based on a local search approach that leads a heuristic search to find an optimal solution by using flexible memory (15). The Tabu Search approach iteratively finds the neighborhood of the present solution which is an initial solution, and the next move has chosen after checking all other solutions. This iterative process might repeatedly execute in a cycle. TS approach uses memory function to avoid the cycling process and to store a tabu list.

The tabu list contains the new solution which is called a tabu solution, and it is frequently looking for the best solution until the search space comes to an end and finds the best optimal solution (16). Besides, TS avoid searching for a duplicated solution within the data structure or domain is known as tabu restrictions.

Algorithm: 1 Tabu Search Pseudo code

```
// initial steps considered as an initial solution
bestSolution => b0
candidate => b0
list => []
/* pushed the initial solution into the list */
list.push(b0)
/* specified condition for stopping the loop by a user */
while (stoppingCondition())
  /* get a list of candidates (neighbourhood) */
  setNeighbourhood => getNeighbours(candidate)
  /* loop through the neighborhood candidates by checking the duplicated
   entries in the tabu list */
  for (searchCandidate in setNeighbourhood)
    if ( (not list.contains(searchCandidate)) and (fitness(searchCandidate)
   > fitness(candidate)) )
      candidate => searchCandidate
    end
  end
  /* if best solution found then push it into the list */
  if (fitness(candidate) > fitness(bestSolution))
    bestSolution => candidate
  end

  list.push(candidate)
  /* if the size of the candidate tabu list is greater then the actual best
   solution then remove it from the list */
  if (list.size > maxTabuSize)
    list.removeFirstItem()
  end
end
return bestSolution
```

2.2.4. Genetic Algorithm

Mitchell et al., (17) has introduced a Genetic Algorithm (GA), which relies on theoretical search approaches and based on the processes of Darwin's theory of evolution. Many researchers had developed the optimization algorithms based on genetic and natural selection approach (18), before J. H. Holland. As mentioned above, GAs were introduced by Holland, and subsequently, it was broadly used in the various field to solve the optimization problems, such as scheduling (19), engineering sector (20), astrophysics (21), and so on.

Genetic Algorithm (GA) derived from the branch of biology which deals with a function of macromolecules, i.e., nucleic acids and proteins. The performance of GA solution is based on the structure of chromosomes that contains good and bad phenotypes or individuals, and the useful phenotypes consider as an optimal solution that uses the natural selection, reproduction and mutation mechanism. In general, the selection of a set of chromosomes to reproduce is based on iteration. Nevertheless, the offspring are produced by crossing the

selected chromosomes, and then the mutation has applied to children chromosomes with the help of a mutation operator. After fulfilling the complete mechanism, the chromosomes can be inserted into the population, but after calculating the quality (fitness) of the solution. Traditionally, the representation of a solution is in the binary form of 0s and 1s, but according to (22), there are other possible encodings too.

Furthermore, GAs are a population-based heuristic approach that starts picking the initial population as a random solution to the given problem. Every random solution throughout the population is known as an individual, and individuals are explicitly assessed to the problem which is called a fitness function.

Algorithm: 2 Genetic Algorithm Pseudo code

```
Initialize population
Assess individual within the society

while ( stoppingCondition() )
    Selection of parent chromosome
    Create offspring by crossover the parents
    Mutation applied to children chromosomes
    Replace old (bad) phenotypes with new generation from the population
end while
```

Later, the selection operator starts choosing the right individuals for the recombination process, and mutation and crossover operators are used in the recombination that creates new individuals to investigate the solution. The old unfortunate individuals have been replaced by new individuals from the population based on their quality (fitness). As mentioned, GAs are executed in the iteration, so once the stopping condition is reached, the evaluation of an algorithm on the population is stopped.

2.3. Technical Approach of Proposed System

The proposed system is based on a web interface and the objective of developing the web-based application for solving the timetabling and scheduling problems by using the combination of tabu search and genetic algorithm to provide a generic interface to the end users. The system is a client-server architecture including different access layers such as application, presentation and database layers. According to (23), this research area is a decision support system which is based on mathematical programming approaches. The system produces optimal timetables by satisfying the constraints on a web-based platform. The main goal of the proposed system is to meet the users and resources or participants (i.e., students, sports trainee, nurses) availability to the course time slots. The algorithm must generate a weekly based timetable for users and participants. In addition, the system can create a group of participants and allocate a slot to them automatically. Aforementioned, there are some tools and approaches in order to solve the timetabling and scheduling problems, but yet insufficient in finding an optimal solution for different required scenarios and use cases. While considering those tools and approaches, this research-based thesis has taken some critical use cases and problems to be able to find a

robust optimal solution. The following possibilities are marked as necessary in timetabling and scheduling problem by satisfying some general soft and hard constraints, which are:

- All resources have to be scheduled and must be satisfied.
- One time slot can be allocated to one resource and multiple units such as students, customers, users, etc. cannot schedule at the same time slot.
- The resource cannot be allocated to the slot where the resource is not available.
- All resources have to be fulfilled the workload weekly.
- The low number of availability of a user has to be scheduled first on the calendar.
- Merge the availability of multiple users in order to make a group unit.
- If there is an empty availability of a resource than for particular resource the scheduling is not possible
- Allow end user to Drag and Drop units to the calendar manually.

By implementing the proposed approach, we can develop a generic optimal robust solution for more than one real-world timetabling and scheduling problems. We can set a benchmark for other researchers in this area of the field, who are willing to contribute to making an optimal solution. In addition, the system must be capable of solving the problem with less computational time and maximize performance. Some features will be produced at the end of this thesis which is as follows:

- Conflict-free, robust and scalable timetables
- Automated time slot allocation
- Availability of users are taken into account
- Intuitive and elegant user experience
- User-friendly timetables

2.4. Architecture

The system architecture design consists of different modules as seen in Fig. 2 such as client, server, algorithm, database, and cache.

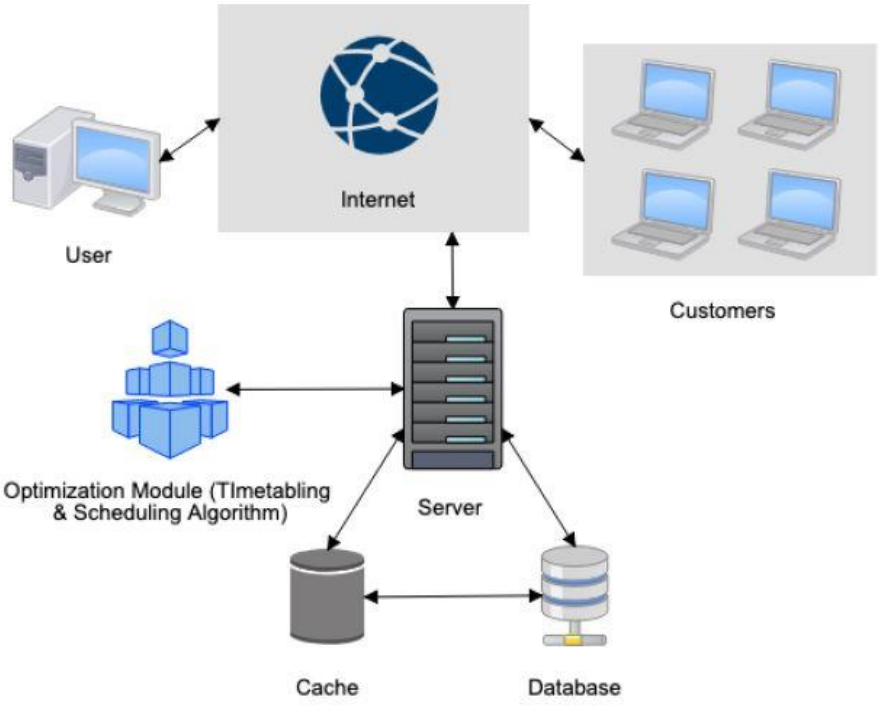


Fig. 2. Architecture design diagram

The client-side represents a web page application which provides the user interface of the system. In the user interface, users are allowed to enter their availability, courses information and other parameters and constraints of the problem. Users can view the timetable on a calendar, import data, and availability, export timetable as well as drag and drop the event within the calendar, and timetable can be saved. The user interface is developed mostly on jQuery 1 library and other JavaScript libraries to build some web components such as Full Calendar 2, Data Tables 3, Draggable 4, etc. The two-way communication with the server is based on AJAX 5. The second module is a database model which stores the information that needs to create timetables. The database which is used in this system is MySQL 6 which is a relational database. MySQL provides the high availability, performance, and scalability for retrieval and storage of the data which makes the data integration faster and easier.

The third module is the cache module which is contributing a big part in the application. There are two types of cache that have been used so far; one is on the client side, i.e., local storage or browser caching and another one is on the server side that holds or stores the data. Whenever data is requested from client-side, the cache module responds instead of requesting the database module all the time. The caches being used to faster the response with less computation. The fourth is an optimization module that contains the algorithm based on the combination of genetic algorithm and tabu search, which is used to create an optimal solution (timetables) for various real-world problems.

In the end, the fifth module is the server side module which is implemented on a framework of PHP 7. The PHP code is general purpose and programming language which embedded in HTML (24), Using a PHP framework provides the scalable and fast web

applications, and also helpful in creating representational state transfer (RESTful) API which is used to process the requests and responses to the application.

3. METHODOLOGY

It will describe steps to solve real-world timetabling and scheduling problems. Furthermore, the fundamental idea of developing the algorithm is from the meta-heuristic approaches. How meta-heuristic algorithm produces an optimal solution; therefore, the

construction of a proposed algorithm is based on those approaches, but, while making the algorithm, we might have used different terminologies and combine different meta-heuristic algorithms such as tabu search and genetic algorithm.

3.1. Solution Steps & Representation

The algorithm to produce an optimal solution for timetabling and scheduling problems are based on the set of parameters and decision making on those parameters. The algorithm must be divided and explained separately in various steps because the algorithm depends upon multiple parameters and constraints that need to be taken into account. The solution is a combination of tabu search in which the list of the population makes a new list of a feasible solution and genetic algorithm which is based on the selection of chromosomes and generation of new chromosomes based on the solution by using crossover, mutation and migration techniques. There are several prerequisites, which need to be finalized in the first place, to execute the algorithm. These preambles are set as input and constraints to the algorithm. This section will briefly explain the solution steps to store and retrieve those points using flowchart diagram and example. Furthermore, the solution contains many steps to fulfill the constraints and find a feasible solution by reducing time consumption. Therefore, it is vital to split the solution steps by representing the flow of each phase, and by breaking the steps, we may create sub-problems from the original problem which is allocating the free time slots to the resources. However, the steps which are necessary to execute the algorithm are as follows:

- How to create a resource and save that particular resource into the system?
- Once the resource has been saved, the way to ask for the resource and storing the availability in the database.
- A possible way to display and retrieve the availability of chunks.
- Resources availability can be updated within the course.
- What are the requirements for creating a course?
- The default length of each unit within the course can be changed by individual length.
- What are the challenges and possibilities to make a group of resources, and how to store and retrieve them from the database?
- Is it essential to store everything in the database, or there is another approach to increase the performance of the system?

3.2. IMPLEMENTATION

According to the below Fig. 3, there are various block that represents the retrieval of information such as resources availability, course available slots, group of resources availability and length of each resource i.e., the duration of resource that will be allocated to the time slot. Once the above information is fetched then we will discuss each section separately. Therefore, the following points may describe the flow of implementation which is as follows:

- Course Slots are a raw data that we need to generate a proper set of course slots, where resources will take place.
- Retrieving the unit availability from single and group of resources availability by merging them.
- Need to normalize the length of each unit (single or group of resources).
- The availability enlistment block represents the reformation of unit availability.
- Correlation and sequencing of units are based on each unit availability in comparison with length of that particular unit.
- Mapping takes place with the generated course slots and sequenced units.
- Once the mapping is covered meaning the algorithm has decided, where to schedule the unit on the free time slot, the boxing further divided the final array into pieces by moving each unit to each box as per unit's available ranking and counts.
- There are only possible three boxes, the box 1 contains all units whose count is just one, meaning the unit is feasible to be allocated for once, box 2 is based on two counts of availability and box 3 will contain the units which have three or more possibilities to be allocated on the calendar.
- At the end of the algorithm, the shooting order takes place which will check if the unit can be scheduled on the timetable or not, if in case the slots is not free or occupied then algorithm will move that specific unit to the outside lane, from where user is able to drag and drop unit on the calendar manually.

Furthermore, there is one more block that indicates the patching of chunks which is a course slots. Before allocating the slots to the units, the course slots are occupying the slots on the calendar with a gray color to represent the allocation will take place within this frame, and no unit will be allocated outside the course slots as well as user cannot place any unit outside the course slot by drag and drop.

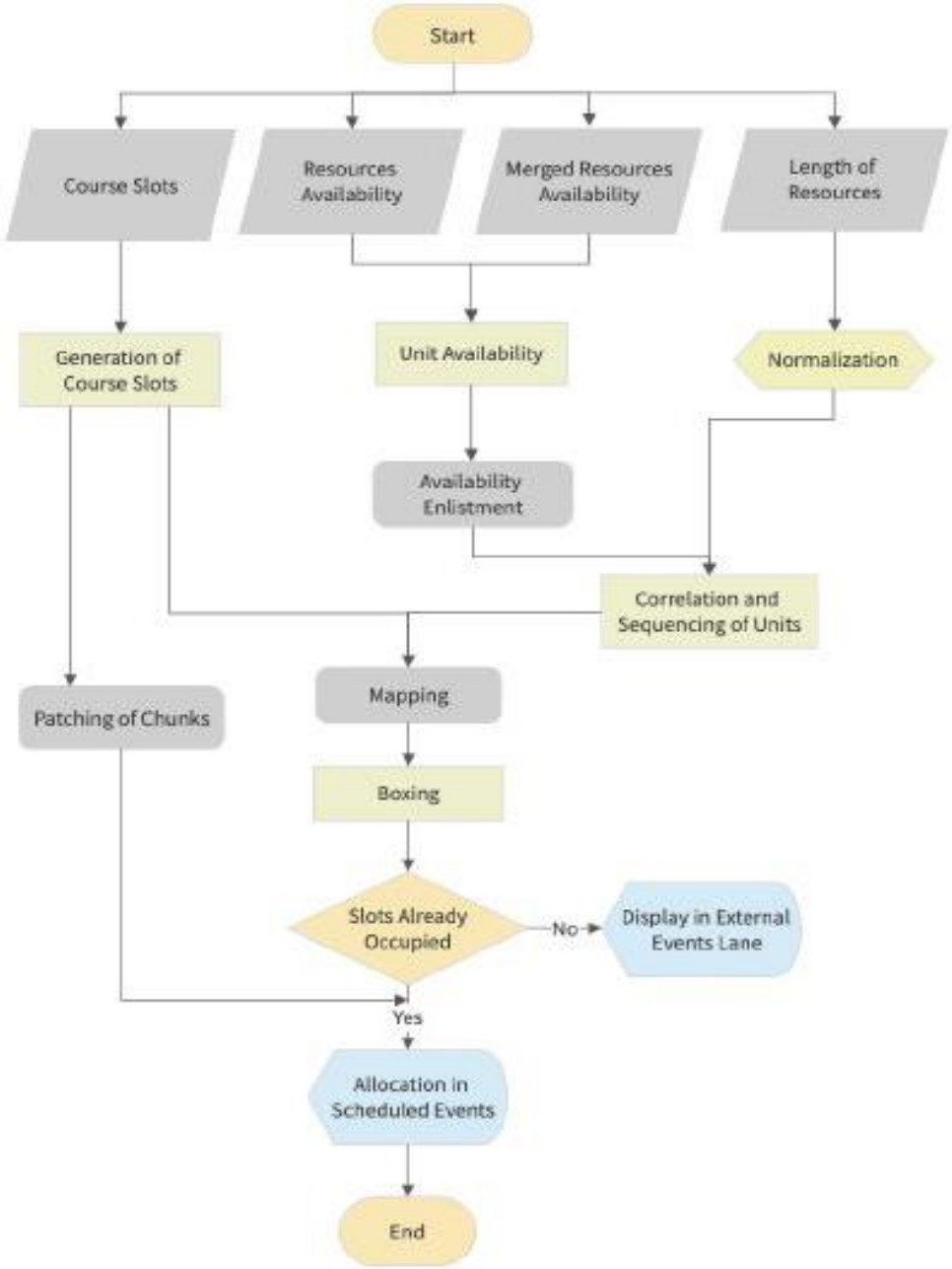


Fig. 3. Flow chart diagram of algorithm overview

4. RESULTS AND FINDINGS

This section represents the experimental results of the proposed algorithm which is developed to satisfy different constraints and allocate time slots to multiple resources. Mostly, algorithm has generated the feasible solution, but in some cases due to the conflicts between resources or lack of availability of a resource ends in no feasible solution for that particular resource, and for those resources, the proposed algorithm provided drag and drop functionality.

4.1. Measures and Performance

This section determines the measurement and performance for the allocation of resources and the combination of availability of resources to time slots. According to the given below Table 2, the performance has measured in time that how long algorithm took to compute and allocate time slots to resources.

Table 2. Performance of resource allocation based on computational time

Number of Resources	Same Resources Lengths	Multiple Resource Available Days	Different Resource Lengths	Different Resource Lengths and Multiple Days
1	0.2s	0.41s	0.561s	1.67s
5	1.41s	1.54s	2.49s	3.44s
10	3.11s	4.69s	5.67s	5.61s
50	6.31s	7.11s	8.15s	9.14s
100	9.37s	10.41s	11.73s	14.11s
150	14.19s	14.79s	16.96s	19.01s
200	17.51s	18.01s	21.11s	23.76s
250	23.21s	23.97s	25.61s	31.16s
300	26.58s	29.19s	32.11s	38.11s
350	29.24s	31.11s	36.11s	42.42s

Fig. 4 shows the performance in line graph that based on the experimental results of the resources. There are four key parameters as per Table 2 such as all resources contain same lengths, resources are available on multiple days, different lengths of resources, and different resource lengths along with multiple days on which resources are available.

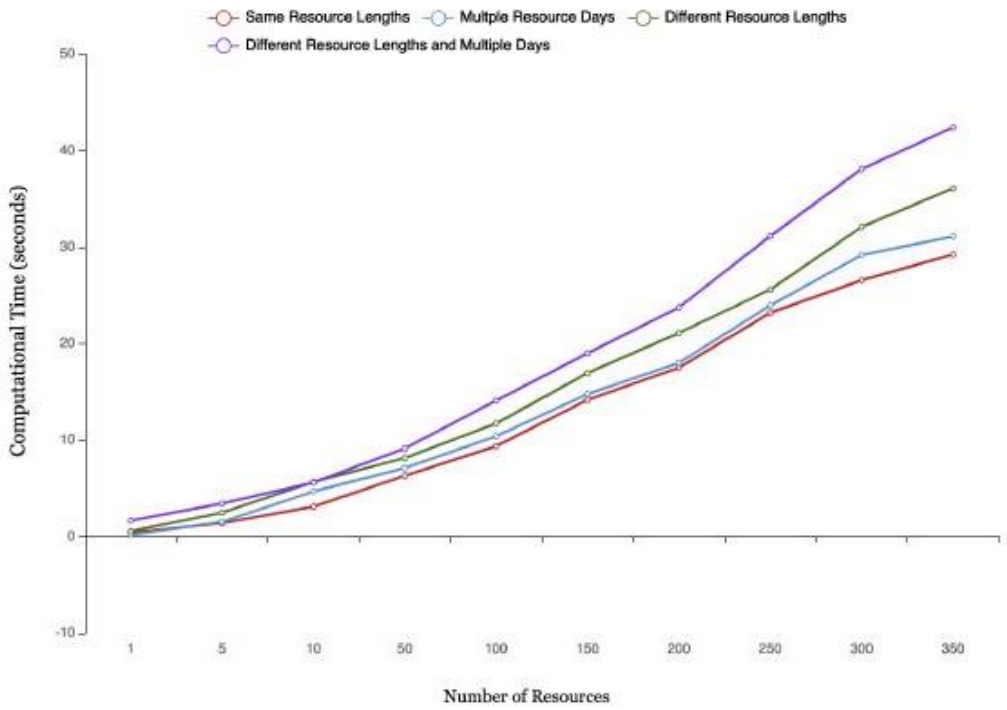


Fig. 4. Performance based on the experimental results of resources

However, with the help of the above graph, we can clearly see the performance of different scenarios of allocation of resources. But, the fundamental concept of measuring the

performance is if the number of resource increases, the computational time will also increase. Basically, the graph illustrates that the number of resources and computational time of the algorithm is directly proportional to each other.

The red line in the graph illustrates that every resource has the same length. For instance, if the length is 30 minutes that means all resources have the same length of duration on which they might schedule. The second line is in blue that shows the resources are available on multiple days, but also possible that one resource is available on Monday and the second resource is available on Monday and Tuesday.

Furthermore, there are two more lines in the graph; therefore, the green line indicates the computational time based on different lengths for resources. For example, if the first resource may schedule on 30 minutes slot then another resource contains 90 minutes of length. In Fig. 4, the purple line represents the last scenario which means the resource may contain different lengths and also available on different days in the week.

5. CONCLUSION

In the justification of an algorithm that we developed in this paper for solving the optimization problem using mathematical computational programming, we have discussed several systematic approaches and parameters and have analyzed the results experimentally in order to maximize the performance of the algorithm. We have also discussed other methods to solve the timetabling and scheduling problems in the literature review and benchmark instances section in detail, but we have seen that the other approaches are mostly specific to resolve an individual problem. Therefore, the model that we designed is more concerned with solving various real world problems based on several sets of requirements. In our model, we have combined the meta-heuristic approaches such as tabu search and genetic algorithm to find an optimal solution for different parameters and constraints. There are many researchers involved in developing such a tool to solve these problems. Although, we have not found any generic tool which might solve or cover more problems in just one algorithm; therefore, we don't have any generic benchmark model. Moreover, we can consider this model as a generic benchmark for different problems. During the development of this model, we came to know about a lot of algorithms for the assignment of resources and finding an optimal solution, then based on the experimental results and performance, we realized if we combined the approaches of two different algorithms then the result will be more optimal. Therefore, we picked the methods from genetic algorithm to map the resource on the time slots based on different population size and tabu search for breaking the availability of resources and course slots and made a new population by reducing the old population, the developed algorithm provided possible results.

However, the complexity of the algorithm is already too high, but in case, if we add more constraints to the algorithm, the complexity will increase more. In this paper, we have investigated with three different sets of parameters only on serial computer to check the results, and the experimental results were not that bad. Therefore, the recommendation for improving the performance of the algorithm, if we check the same experiment on

parallel or distributed computers then the experimental results will remain the same but the performance will be increased, and the complexity level will be divided to computers. Furthermore, the paper has stated on various techniques, and some experimental results represent the potential usage of combining the basic concept of meta-heuristic approaches to solving real-world timetabling and scheduling problems. Within this paper, we have studied and performed some experiments on different parameters and constraints, in order to obtain a feasible and robust solution. Based on the analysis and performance of the algorithm, we can say the results were promising, and this paper can lead to many future directions.

5.1. Future Enhancements

There are various possibilities to enhance this system; the first aspect is to introduce more constraints on the algorithm and find the best possible solution. Usage of distributed servers and high-performance data storage and cache can increase the efficiency of the algorithm, which will also minimize the computational time. Introduce artificial intelligence in the algorithm to collect availability from resources based on some patterns stored in the system.

Reference:

1. Schmidt G, Ströhlein T. Timetable construction – an annotated bibliography. Comput J. 1980 Jan;23(4):307–16.
2. Abdennadher S, Marte M. University course timetabling using constraint handling rules. Appl Artif Intell. 2000 Apr;14(4):311–25.
3. Bellio R, Di L, Andrea G. A Statistical Analysis of the Features of a Dynamic Tabu Search Algorithm For Course Timetabling Problems. 2007;1–3.
4. Elmohamed MAS, Coddington P, Fox G. A Comparison of Annealing Techniques for Academic Course Scheduling. 1998;92–112.
5. Abdullah S, Turabieh H. Generating University Course Timetable Using Genetic Algorithms and Local Search. In: 2008 Third International Conference on Convergence and Hybrid Information Technology. 2008. p. 254–60.
6. Al-Betar M, Khader AT. A hybrid harmony search for university course timetabling. Proceedings of the 4nd Multidisciplinary Conference on Scheduling: Theory and Applications (MISTA 2009). 2009.
7. Abdullah S. HEURISTIC APPROACHES FOR. 2006;(June).
8. Burke E, Jackson K, Kingston JH, Weare R. Automated University Timetabling: The State of the Art. Comput J. 1997 Jan;40(9):565–71.
9. Baker KR. Workforce Allocation in Cyclical Scheduling Problems: A Survey. J Oper Res Soc. 1976 Apr;27(1):155–67.
10. Solos IP, Tassopoulos IX, Beligiannis GN. A Generic Two-Phase Stochastic Variable Neighborhood Approach for Effectively Solving the Nurse Rostering Problem. 2013;278–308.
11. Ncibi K, Sadraoui T, Faycel M, Djenina A. A Multilayer Perceptron Artificial Neural Networks Based a Preprocessing and Hybrid Optimization Task for Data Mining and Classification. 2017;5(1):12–21.
12. Osman IH, Kelly JP. Meta-Heuristics: An Overview BT - Meta-Heuristics: Theory and Applications. In: Osman IH, Kelly JP, editors. Boston, MA: Springer US; 1996. p. 1–21.

13. Glover F. PATHS FOR INTEGER PROGRAMMING. 1986;13(5):533–49.
14. Glover F, Laguna M. Tabu Search BT - Handbook of Combinatorial Optimization: Volume1–3. In: Du D-Z, Pardalos PM, editors. Boston, MA: Springer US; 1998. p. 2093–229.
15. Burke EK, Kendall G, Soubeiga E. A Tabu-Search Hyperheuristic for Timetabling and Rostering. J Heuristics. 2003;9(6):451–70.
16. Glover F, Laguna M. General purpose heuristics for integer programming—Part I. J Heuristics. 1997;2(4):343–58.
17. Mitchell M, Holland JH, Arbor A. When Will a Genetic Algorithm Outperform Hill Climbing ? { I.
18. Bremermann H. The evolution of intelligence : the nervous system as a model of its environment. Seattle, Wash.: University of Washington, Dept. of Mathematics; 1958.
19. Hertz A. Finding a feasible course schedule using Tabu search. Discret Appl Math. 1992;35(3):255–70.
20. Altshuler EE, Linden DS. Wire-antenna designs using genetic algorithms. IEEE Antennas Propag Mag. 1997;39(2):33–43.
21. Charbonneau P. Genetic Algorithms in Astronomy and Astrophysics. Astrophys J Suppl Ser. 1995 Dec;101:309.
22. Whitley D. A genetic algorithm tutorial. Stat Comput. 1994;4(2):65–85.
23. Liang T-P, Lee C-C, Turban E. Model Management and Solvers for Decision Support BT - Handbook on Decision Support Systems 1: Basic Themes. In: Burstein F, W. Holsapple C, editors. Berlin, Heidelberg: Springer Berlin Heidelberg; 2008. p. 231–58.
24. Lerdorf R. Introduction to PHP Rasmus Lerdorf. 2003;