a JES Journal of Engineering Sciences

Vol 13, Issue 06, June/2022
ISSN NO:0377-9254

Survey on Timetable Generator for a College or University Department

K Haripriya [1]
Assistant Professor
Computer Science and Engineering
VNR VJIET
Hyderabad, India
haripriya_k@vnrvjiet.in

MD Abdul Rab [2]
B-Tech Graduate
Computer Science and Engineering
VNR VJIET
Hyderabad, India
mohdrab01@gmail.com

N Venkata Bharadwaja [3]
B-Tech Graduate
Computer Science and Engineering
VNR VJIET
Hyderabad, India
bharadwajlive@gmail.com

L Goutham Reddy [4]
B-Tech Graduate
Computer Science and Engineering
VNR VJIET
Hyderabad, India
gouthamlyagala@gmail.com

K Saicharan [5]
B-Tech Graduate
Computer Science and Engineering
VNR VJIET
Hyderabad, India
saicharank018@gmail.com

***Abstract*— Our aim is to come up with a feasible solution for creating a schedule in a university department. We have taken up this problem since we are aware that it is a persistent problem in educational institutions. We'll use genetic algorithms to tackle a complex timetabling problem for a university or college. In order to get workable timelines in a fair amount of time, heuristics and context-based reasoning were applied. Apart from providing a feasible solution, we will also be developing a UI that will be helpful for the college administrators to set up constraints in real-time. They can add the available subjects, faculty, no. of hours, and all the required data using the UI. Then the algorithm would be run at the backend and generate the time tables which also would then be rendered in the UI.**

Keywords*— *Timetable, Hybrid, Genetic, Heuristic, Constraints, UI.

1. INTRODUCTION

Colleges offer diverse courses, professors deal with different courses in different semesters, and will handle multiple subjects, all of which take a lot of time and must be completed without overlap. Because effective time management is critical, we want automatic timetable scheduling. Manual scheduling is difficult, especially when the amount of data and resources to manage grows and the risk of collisions increases.

To provide a roadmap for this paper, the first section contains an introduction to the paper, followed by related work in section 2, the timetabling problem in section 3, approaches to timetable generation in section 4, a few timetable generation algorithms and their challenges in section 5, and conclusion in section 6.

2. RELATED WORK

The metaheuristic Algorithmic, according to Abeer Bashab Et. Al, works better because it uses current information obtained by the algorithm to better determine which alternative solution should be examined next, or how the next candidate should be developed.

They used the Genetic Algorithm, which is a subset of Evolutionary Computation, a broader area of computation, according to Swapnil Markal et al. The disputes between the subjects are handled adequately in compared to other systems.

They employed a typical backtracking method, which is a brute force tactic, according to V. Abhinaya Et. Al. It takes a long time to calculate and is only useful for one type of class.

Meta heuristic-based approaches are common in handling the university scheduling problem, according to Mei Ching Chen Et.Al, and hybrid methods are close behind. On the other hand, hyper-heuristic procedures are also effective.

They devised a genetic algorithm that was acceptable in speed and converged faster than earlier fundamental algorithms, according to Shraddha Thakare et al. It's also worth noting that this model can be enhanced by including other changes.

Mr. Mallikarjuna Nandi Et.Al suggested an optimization approach to improve search efficiency by combining different strategies. It also covers a few key hard constraints, such as faculty availability conflicts, as well as non-rigid soft constraints, such as search technique optimization goals.

To solve the timetabling problem, Amin Rezaeipanah et al proposed a technique that combines the Improved PGA Algorithm with Local Search. This technique has been tested against a variety of benchmarks, and the findings indicate that it is more efficient and effective than other typical UCTP approaches.

Cheng Weng and colleagues created a version of the approach that increases global exploration by combining a global best model prompted by particle swarm optimization with the great deluge strategy to encourage local exploitation. This strategy achieves a fair mix of exploration and exploitation.

Shraddha Shinde et al. demonstrated that scheduling lectures with a genetic algorithm is both efficient and beneficial. They have demonstrated a high possibility for a future leadership timetable that is more equitable to students using the technique they have proposed. The framework is

straightforward to use and appears to be directly applicable to a wide range of scheduling issues.

According to H. Algethami et al., they used a mixed integer programming technique to address the university scheduling problem. Faculty preferences, availability, and timeslot and room assignment limits were all taken into consideration. They wanted to get the most out of faculty assignments and activities by tailoring them to their specific needs.

They have attempted to detect whether a certain timetable is viable or not based on the presence of binary session conflicts, according to David Schneider et al. Second, they wanted to test if the existing ProB tools could be utilised in a larger project to run models in a production system.

According to Tanzila Islam et al., they used the scoring concept, in which they tried to come up with the best routine depending on the score. They used the Tabu Search concept, which is a meta-heuristic for addressing optimization problems. The Tabu Search approach produces an unsatisfactory first result. Tabu Search provides a reasonable response in a fair amount of time.

Evolutionary techniques were used to tackle the university timetabling problem, according to Mayuri Bagul et al. Evolutionary algorithms, genetic algorithms, and other methodologies have been used, with mixed outcomes. In their research, they used a genetic algorithm to solve the problem. To solve the problem, they used a mimetic hybrid, genetic artificial immune network algorithm, which they compared to the results of the Genetic Algorithm. The genetic artificial immunity network outperforms the Genetic Algorithm in terms of the best option.

3. TIMETABLING PROBLEM

Scheduling various tasks is a prevalent decision challenge in most businesses or institutions, particularly in universities and colleges. The university course timetabling challenge is a complex need to allocate certain activities at the university or college, such as Courses, professors, and pupils are all offered in a set number of time slots. However, in any institution, organisation, or university, the timetabling difficulty can be adjusted according to events as follows:

1. Education-related timetabling
2. Sports-related timetabling
3. Employment-related timetabling
4. Transport related timetabling, etc.,

Throughout this work, we have concentrated on educational timetabling. It's a complex combinatorial problem with both soft and hard constraints. University course scheduling problem is an NP hard problem, which means it cannot be solved in polynomial time as the size and complexity of the problem grows exponentially.

4. TIMETABLE GENERATION APPROACHES

The following are the several types of timetable generating algorithms:

1. Operational Research-based approach
2. Single solution based Meta Heuristic approach
3. Population based Meta Heuristic approach
4. Hyper Heuristics approach
5. Multi objective approach
6. Hybrid approach

Let us see about each of these in some detail.

4.1 Operational Research based approach

Graph colouring is commonly used in OR (Operational Research) approaches. The graph colouring task involves assigning least no of colours to all of a graph's vertices, with the vertices connecting edges having distinct colours. The current scheduling problem and the graph colouring difficulties are connected in that the events represent the graph's vertices,the edges of the graph are represented by collisions between events, and the time slots are the colours to be assigned. Least Saturation Degree First (LSDF), Integer Programming (IP) and Mixed Integer Linear Programming (MILP) and others are examples of OR-based algorithms.

4.2 Single solution based Meta-Heuristic approach

Local search algorithms are meta heuristic algorithms with only one solution.The term "local search algorithms" refers to algorithms that start with a single solution and then look for a better one in the neighboring regions. This approach's algorithms can be further divided into three categories of optimization algorithms:

One-step optimization techniques satisfies both hard and soft constraints.

Two-stage optimization methods evaluate only hard constraints in the first phase and just soft constraints in the second phase of generating a viable solution.

Relaxation-based algorithms allow for two sorts of relaxation. First, put away any events that cannot provide a viable solution, and then build dummy events to generate viable solutions. Simulated Annealing (SA), Iterated Local Search (ILS) , Tabu Search (TS) , Hill Climbing (HC), and others are some of the algorithms used in this technique.

4.3 Population based Meta-Heuristic approach

As the name implies, these algorithm operate on a population of solutions. They generate a new population of solutions in the neighbourhood of current solutions using a range of operators and rules. They are made up of genetic algorithms that generate an initial population by assigning classes to periods at random, taking into account the room's capacity. Then, using genetic operators like mutation and crossover, the fitness values of parents would assist in choosing their selection in the generation of children for the next generation. In the current timetabling situation, this representation could actually avoid the conflict between the courses. Genetic Algorithms , Ant Colony Optimization, and others are examples of population-based meta heuristic algorithms.

4.4 Hyper-Heuristics approach

Hyper heuristic techniques utilise many heuristics in an adjustable manner to solve the problem. These are usually

simple and quick algorithms that can tackle a range of problems and are adaptive to different instances of specified benchmark datasets. Hyper-heuristics are algorithms that search a space of heuristics rather than a space of solutions in order to find heuristics. This methodology is demonstrated by the Add-Delete Hyper-Heuristic method, which employs an adaptive heuristic generating method to generate a variable-sized list of add and delete operations.

4.5 Multi-objective approach

The algorithms for the multi-objective techniques were created with either a single or several interruptions. The term single-disruption refers to the disruption of one lecture, whereas many-disruptions refers to the disruption of multiple ones. The algorithm that used numerous disruptions outperformed the one that used only one. Multi-Objective Simulated Annealing (MOSA) is an example of this type of technique, with the goal of defining a good Pareto front that considers both solution quality and robustness.

4.6 Hybrid approach

The hybrid approaches can simply be defined as the usage of the combination of other different approaches in order to get rid of previous disadvantages and gain some efficiency as well. A few algorithms under the hybrid approach are as follows: Adaptive Tabu Search (ATS), Hybrid Genetic Algorithms (HGA), Greedy Randomized Adaptive Search Procedure (GRASP), etc.

5. FEW ALGORITHMS

5.1 Mixed Integer Programming

In general, the MIP based model describes the assignment of a particular course to a specific room in a specific time slot as a binary decision variable with penalties if the courses are assigned to overlapping time slots.

Below are a few formulae that would be used in a MIP based model to solve a timetabling problem. The terms used in the formulae are also described as below :

$$\min \quad \sum_{i\in E}\sum_{c\in C_i}\sum_{r\in R_i} \alpha_{icr}x_{icr} + \sum_{i\in E}\sum_{j\in E_i}\sum_{c\in C_i}\sum_{d\in C_j} \beta_{ijcd}\,y_{ijcd} + \sum_{(c,d)\in S}\sum_{r\in R} \gamma z_{cdr}$$

s.t.

$$\sum_{c\in C_i}\sum_{r\in R_i} x_{icr} = 1, \quad \forall i \in E$$

$$\sum_{r\in R_i} x_{icr} + \sum_{r\in R_j} x_{jdr} - y_{ijcd} \le 1, \quad \forall i \in E,\ j \in E_i,\ c \in C_i,\ d \in C_j$$

$$\sum_{i\in E_{cr}} x_{icr} + \sum_{i\in E_{dr}} x_{idr} - z_{cdr} \le 1, \quad \forall (c,d) \in S,\ \forall r \in R$$

$$x_{icr} \in \{0,1\}, \quad \forall i \in E,\ \forall c \in C_i,\ \forall r \in R_i$$

$$y_{ijcd} \in \{0,1\}, \quad \forall i \in E,\ j \in E_i,\ c \in C_i,\ d \in C_j$$

$$z_{cdr} \ge 0 \text{ and integer}, \quad \forall (c,d) \in S,\ \forall r \in R$$

where,

E = the set of courses
Ei = the set of courses that have non-zero conflict severity with course i
Ecr = the set of courses for which color c and room r are suitable
Ci = the set of acceptable colors for the course i
Ri = the set of acceptable rooms for the course i
αicr = inherent penalty of assigning a color (timeslot) c and room r to the course i
βijcd = penalty for assigning course i to color c and course j to color d
γ = penalty for double-booking a room
xicr = 1 if course i is assigned color c and room r, 0 otherwise
yijcd = 1 if i is assigned color c and j is assigned color d, 0 otherwise
zcdr = number of times room r is assigned overlapping colors c and d

And the **one-pass scheduler** algorithms is as follows:
Input :
G the uncolored weighted graph representing the timetabling problem
Hv the vertex-selector heuristic parameters
Hcr the color-selector and room-selector heuristic parameters

while there are uncolored vertices **do**
  $v \leftarrow selectVertex(G, H_v)$
  $(color, room) \leftarrow selectColorAndRoom(v, G, H_{cr})$
  assign $(color, room)$ to v

5.2 Tabu Search

A Meta heuristic optimization search strategy that uses local search methods to accomplish optimization. It works with the sub-optimal initial solution.

It ignores the unessential exploration and analyses the search space upon which optimization is performed and keeps note of the recently visited area in the Tabu list.

A hill-descending heuristic is guided to continue research by the concept of memory structures. In tabu search, there are three different types of memory structures:

- Short term: This contains the list of solutions that are recently considered.
- Intermediate term: This contains rules that guide the search space towards promising areas.
- Long yterm: This contains rules that guide search into new regions.

1. Create an initial solution that is selected at random and assign it as the best solution.
2. Iterate until you find the best solution.
3. If the best local has a higher fitness value than the existing best, update as the best solution.
4. Add the local best candidate to the list and if the list is full, some elements will be removed from the list.
5. The order of expiration of elements in the tabu list is the same order in which the elements are added.
6. Keeps track of the best solution in the neighbourhood until one of the two requirements is met, which is either a fitness score or a time restriction is exceeded.

7. Provide the best solution discovered during the procedure.

```
sBest ← s0
bestCandidate ← s0
tabuList ← []
tabuList.push(s0)
while (not stoppingCondition())
   sNeighborhood ← getNeighbors(bestCandidate)
   bestCandidate ← sNeighborhood[0]
   for (sCandidate in sNeighborhood)
      if ( (not tabuList.contains(sCandidate)) and (fitness(sCandidate) >
fitness(bestCandidate)) )
         bestCandidate ← sCandidate
      end
   end
   if (fitness(bestCandidate) > fitness(sBest))
      sBest ← bestCandidate
   end
   tabuList.push(bestCandidate)
   if (tabuList.size > maxTabuSize)
      tabuList.removeFirst()
   end
end
return sBest
```

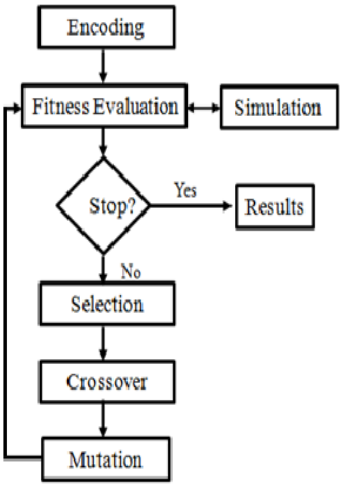
5.3 Hill Climbing

The following is the Hill Climbing (HC) optimizer algorithm:

ALGORITHM :

1. Input(x'){Initial solution}
2. While (Local optimal solution is not reached)d0
3. x" =Explore(N(x'))
4. if(f(x")<f(x'))then
5. x'=x"
6. end if
7. end while

5.4 GeneticAlgorithms



A genetic algorithm is an optimization approach that attempts to identify the best solution based on the input data. The fittest individuals will survive in subsequent generations and will be picked for reproduction in order to generate offspring for the following generation, according to this algorithm. The algorithm starts with

1. Creating a random initial population which will be the set of all possible solutions.
2. The algorithm then uses mutation and combination to create a new population from the old population.
3. The procedure is carried out across several generations.
4. A fitness value is assigned to each individual solution in the population, and solutions with higher fitness values are more likely to participate in mutation and combination, leading in a better population.
5. In this manner, we continue to "evolve" better persons or solutions with high fitness values across generations until we find the best solution or reach a stopping point.
6. Genetic algorithms start with random solutions and aim to find the best ones in subsequent generations.

$$F(X)= \frac{\sum_{i=1}^{n} X_i * W}{t}$$

The challenges faced during this implementation was

1. we can't assign a single subject continuously.
2. A class cannot be assigned to two distinct lectures at the same time.

5.5 Ant-Bee Colony Optimization

It is a population based stochastic technique which has shown excellent search performance on a range of optimizing problems. This strategy is on the basis of conventional intelligent behaviour of honey bees. The colony of ant-bees is separated into two groups.

1. Employed Bees
2. Un-Employed Bees(onlooker and scout bee

These three types of bees (employed, onlooker,scout bees) cooperate together to search for food.
The Ant-Bee Colony algorithm's general scheme is as follows:

```
Initialization Phase
REPEAT
  Employed Bees Phase
  Onlooker Bees Phase
  Scout Bees Phase
  Memorize the best solution achieved
UNTIL(Cycle reaches maximum number limit or CPU time)
```

Working or phases of the algorithm:

1. The ABC algorithm generates a randomly distributed population of a fixed size during the initialization phase.
2. The employed bees complete the food search, make improvements to existing solutions depending on neighbourhood searches, and evaluate the nectar (fitness) of new food source, all while sharing their findings with other bees.
3. Onlooker bees use information gathered from employed bees to select the greatest food source.

This decision is made using the roulette wheel selection approach.

$$p(i) = fit(i) \div \sum_{j=1}^{N} fit(j)$$

Where,

The probability of selecting food source I is given by p(i).The fitness value of a certain dietary source I is fit (i).

Onlooker bees prefer the food supplies advertised by employed bees with a higher volume of nectar.

4. The food source is abandoned if the quality of the food do not improve after a certain number of cycles. Scout bees find a new food source, which will soon be replaced by an abandoned one..

As a result, the bees work together, communicate and cooperate to locate food source or solutions by analyzing nectar quality.

5.6 Hybrid Harmony Search

The Hybrid Harmony Search Algorithm (HHSA) combines memory with the global best idea. In order to adapt the HHSA to the timetabling problem, the schedule representation (x) and objective function (f(x)) are modelled. After that, the datasets are turned to a useful data structure. Finally, the parameters of the HHSA must be determined.

ALGORITHM :

1. Initialize the problem and HSA parameters.
   - Input the data instance of the optimisation problem
   - Set the HSA parameters (HMCR,PAR,NI,HMS).
2. Initialize the harmony memory.
   - Construct vectors of the harmony memory, HM={x1,x2,x3.....,xHMS}
   - Recognize the worst vector in HM. xworst {x1,x2,....,xHMS}
3. Improvise a new harmony
   x' // new harmony vector
   for i= 1,....,N do
   if (U(0,1) <HMCR)then
   x'i {x1i,x2i,...,xiHMS} {memory consideration)
   if (U(0,1)<PAR)then
   x'i=vi,k+m{pitch adjustment)
   end if
   else
   x'i{random consideration}
   end if
   end for
4. Update the harmony memory
   if (f(x')<f9xworst))then
   Include x' to the HM.
   Exclude xworst from HM.
   end if
5. Check the stop criterion
   while (not termination criterion is specified by NI) do
   Repeat STEP3 and STEP4
   end while

5.7 Steady State Genetic Algorithms

The algorithm used was as follows:

ALGORITHM :

1.Randomly initialize a population of solutions
2.Evaluate the individuals in the population
3.Apply local search to each individual of the population
4 While the termination condition is not reached do
5 Select two parents through tournament selection
6 Create a child by crossover with a probability Pc
7 Apply mutation to the child with a probability Pm
8.Apply local search to the child
9 Replace the worst member of the population by the child
10 End while

6. CONCLUSION

According to a large number of papers in the scientific literature, the university course scheduling problem is an active and essential research topic. The initial appearance of international competitions in timetabling continues to inspire. This research looks at the different strategies presented in the last 10 years to handle the challenge of university course scheduling (both real world and benchmark). The approaches are divided into various groups. They are also organised chronologically by year of publication. As a result, an overview of present trend in this domain is provided. In addition, the limitations of each methodological group are discussed in this study.

ACKNOWLEDGMENT

VNR VJIET provided support for this research. Although they may not agree with all of the interpretations presented in this study, we are grateful to our colleagues who gave expertise that considerably helped the research.

We are also grateful to Mrs. K. Haripriya Madam, our Major Project Guide, for her support and for moderating this paper, which considerably enhanced the document.

REFERENCES

[1] H. Algethami, W. Laesanklang A Mathematical Model for Course Timetabling Problem With Faculty-Course Assignment Constraints August 2021 DOI:10.1109/ACCESS.2021.3103495

[2] MEI CHING CHEN, SAN NAH SZE, SAY LENG GOH, NASSER R. SABAR, GRAHAM KENDALL. A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities July 2021 DOI: 10.1109/ACCESS.2021.3100613

[3] Shraddha Thakare, Tejal Nikam, Prof. Mamta Patil. Automated Timetable Generation using Genetic Algorithm IJERTV9IS07056 Volume 09, Issue 07 (July 2020) IJERT

[4] Swapnil Markal, Surekha Ghorpade, Diksha Chalke. Timetable Generator e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 2, Ser. II (Mar - Apr 2020), PP 29-

[5] Abeer Bashab, Ashraf Osman Ibrahim, Eltayeb E. Abed Elgabar, Mohd Arfian Ismail, Abubakar Elsafi, Ali Ahmed, Ajith Abraham. A systematic mapping study on solving university timetabling problems using meta-

heuristic algorithms. Neural Computing and Applications (2020) 32:17397–17432

[6] Amin Rezaeipanah, Samaneh Sechin Matoori, Gholamreza Ahmadi. A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Applied Intelligence volume 51, pages467–492 (2021)

[7] V. Abhinaya, K. Sahithi, K.Akaanksha. Online Application of Automatic Time-Table Generator. Volume: 06 Issue: 02 | Feb 2019

[8] Mr.Mallikarjuna Nandi, Ms.R.Priyadharshini, Ms.R.Aishwarya and Ms.M.Nandhini. AUTOMATIC TIME TABLE GENERATION. Vol 12 Issue 1 2019

[9] David Schneider, Michael Leuschel , Tobias Witt. Model-based problem solving for university timetable validation and improvement. Formal Aspects of Computing volume 30, pages545–569 (2018)

[10] Parkavi A. A STUDY ON AUTOMATIC TIMETABLE GENERATOR MAY 2018 ISSUE VOLUME 5 ISBN No. 978-81-923607-3-7

[11] Tanzila Islam, Zunayed Shahriar, Mohammad Anower Perves, Monirul Hasan. University Timetable Generator Using Tabu Search DOI: 10.4236/jcc.2016.416003

[12] Cheng Weng Fong, Hishammuddin Asmuni, and Barry McCollum. A Hybrid Swarm-Based Approach to University Timetabling. IEEE Transactions on Evolutionary Computation (Volume: 19, Issue: 6, Dec. 2015) DOI: 10.1109/TEVC.2015.2411741

[13] Dipesh Mittal, Hiral Doshi, Mohammed Sunasra , Renuka Nagpure. Automatic Timetable Generation using Genetic Algorithm Vol. 4, Issue 2, February 2015. DOI 10.17148/IJARCCE.2015.4254

[14] Mayuri Bagul, Sunil Chaudhari, Sunita Nagare, Pushkar Patil. A Novel Approach for Automatic Timetable Generation. Volume 127 – No.10, October 2015

[15] Mohammed Azmi Al-Betar, Ahamad Tajudin Khader, and Munir Zaman. University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm. IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews) 42(5):664-681 DOI:10.1109/TSMCC.2011.2174356

[16] Shengxiang Yang, and Sadaf Naseem Jat. Genetic Algorithms With Guided and Local Search Strategies for University Course Timetabling. IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews) 41(1):93 - 106 DOI:10.1109/TSMCC.2010.2049200.

[17] A systematic mapping study on solving university timetabling problems using meta-heuristic algorithms Abeer Bashab, Ashraf Osman Ibrahim, Eltayeb E. AbedElgabar, Mohd Arfian Ismail, Abubakar Elsafi, Ali Ahmed & Ajith Abraham

[18] AnujaChowdhary “TIME TABLE GENERATION SYSTEM” .Vol.3 Issue.2, February- 2014, pg. 410-414

[19] Antariksha Bhaduri “University Time Table Scheduling using Genetic Artificial Immune Network” 2009 International Conference on Advances in Recent Technologies in Communication and Computing

[20] A. Elkhyari, C. Gu´eret, and N. Jussien, “Solving dynamic timetabling problems as dynamic resource constrained project scheduling problems using new constraint programming tools. In Edmund Burke and Patrick De Causmaecker, editors, Practice And Theory of Automated Timetabling, Selected Revised Papers,” pp. 39–59. Springer- Verlag LNCS 2740, 2003.

[21] DiptiShrinivasan “automated time table generation using multiple context reasoning for university modules” Published in: evolutionary computation, 2002. cec '02. Proceedings of the 2002 congress on (volume:2)

[22] Anirudha Nanda “An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach”. International Journal of Machine Learning and Computing, Vol. 2, No. 4, August 2012.

[23] Sadaf N. Jat, Shengxiang Yang “ A mimetic algorithm for university course timetabling problem”, 2008 20th IEEE International conference on tools with artificial intelligence.

[24] Hitoshi kanoh, Yuusuke sakamoto “Interactive timetablng system using knowledge based genetic algorithm” , 2004 IEEE International conference on systems, man and cybernetics.

[25] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee, “A survey of search methodologies and automated system development for examination timetabling,” J. Schedul., vol. 12, no. 1, pp. 55–89, 2009.

[26] B. Paechter, A. Cumming, M. G. Norman, and H. Luchian. Extensions to a memetic timetabling system. Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling, LNCS 1153, pp. 251–265, 1996.