Introducing a Novel Parameter in Generation of Course Timetable with Genetic Algorithm

Ravitashaw Bathla[1], **Shubham Jain**[1] , **Rajeev Singh**[1]

[1] Department of Computer Engineering
G.B. Pant University of Agriculure and Technology
Pantnagar, Uttarakhand, India

{ravitashaw@outlook.com, sj38711@gmail.com, rajeevpec@gmail.com}

Abstract. In this paper, we introduce a new Happiness parameter along with Genetic Algorithm for generating course timetable. This happiness parameter will generate appropriately feasible solution and account for the comfort and happiness of the instructor and students both (indicating the appropriateness of the resulting solution). The final result obtained from this approach shows that the solution space is reduced considerably and hence a feasible solution is obtained. Using this parameter, it can also be analysed that the solution obtained from Genetic Algorithm without Happiness Parameter are unfavourable most of the times. We perform experiments on data of Department of Computer Engineering, G.B. Pant University of Agriculture and Technology, Pantnagar and are able to produce promising results.

Keywords: Genetic algorithm, Timetable, Scheduling, Happiness parameter

1 Introduction

The university timetabling problem can be considered as the task of assigning a number of events, such as lectures, exams; to a limited set of timeslots (and rooms), in accordance with a set of constraints. The constraints vary greatly from university to university, since all have their own specific requirements. The timetabling problem is an NP-Hard problem [1]. This is very difficult to solve using conventional methods or manually and costs a lot of resources, time and money [2].
Among several timetable related problems in academic domain, the Course Timetabling and Exam Scheduling problem are prominent one. The two problems exhibit very similar characteristics and have different requirements depending upon institutional needs. The course timetable problem deals with assigning one event per room per timeslot and involve allocation of events to a fixed set of timeslots. The exam scheduling problem involves multiple event scheduling in a single classroom at the same time (provided seating capacity constraints are not violated) and allow flexibility in number of timeslots. In this

paper, we have chosen the Course Timetable Problem as it provides more versatile problem domain and the set of constraints are robust.
The various constraints that might be imposed on a particular timetabling problem also varies according to the level of impact and importance of each of the constraint. The constraints are usually divided into two categories: Hard constraints, and the soft constraints [3]. The priority is given to hard constraints over soft constraints. A timetable is considered feasible if all the hard constraints are satisfied [4] and are usually same for most of the universities. Meanwhile, soft constraints are those that should be obeyed if possible, and vary with universities. Most universities will have their own idiosyncratic set of constraints according to their requirement that make their timetabling problem different from others. These eccentric behavior of soft constraints for each university make it very difficult for the generalization of the problem [5].
We propose to generate an appropriate feasible timetable solution using our new Happiness parameter along with Genetic Algorithm. The Happiness parameter is implemented after the application of Genetic Algorithm, on those solutions that have fitness value of 1.0. Using this parameter, feasible solution(s) is (are) obtained and the solution space is reduced considerably.
The rest of paper consists of 7 sections. Section 2 showcases related work done in the field. Section 3 describes the problem description in detail. Section 4 presents the standard Genetic Algorithm. Section 5 presents the proposed methodology including the new happiness parameter. Section 6 contains results and discussions of the experiments performed and finally section 7 concludes the work done.

2 Related Work

The work done by Bambrick (1997) [6], presents Genetic Algorithm for solution of timetable problem with repair strategy. The final solution is selected stochastically from all the random solutions that have fitness count of 1.0. Sometimes least feasible (non-appropriate) solutions are obtained as final solution because the solution space is wide enough. These non-appropriate solutions also exists with feasible solution, all having fitness 1.0.
According to the work presented by Abdullah and Turabieh (2008) [7], local search is used for optimization of the solution after implementation of Genetic Algorithm. Local search technique optimizes the solution by reducing the number of soft constraints. An additional computational cost is incurred in doing this optimization. The optimization is implemented on the fittest solution in the population which limits the final solution, as final solution is the result of further optimization of that fittest solution.
Yang and hat (2011) [8] have implemented local search and guided search technique along with Genetic Algorithm for solution of the course timetable problem. An additional data structure in guided search technique was used that stores information about the events that have zero penalty value [2]. This data

structure is used every time a new generation is produced for passing of traits to new generations.

3 Problem Description

In course timetable problem, a set of events (Lectures, Practical and Tutorials) have to be allotted to fixed number of rooms and timeslots within a week for instructors and group of students [7].
In this paper, we generate the timetable of Department of Computer Engineering, G.B. Pant University of Agriculture and Technology. The course structure is defined such that each course bears a number of credit hours. The credit hours designates the total number of lectures and practical. The tutorials are not mandatory for all courses. They are not included in the total credit hours. A course is represented as Course Number (Lectures-Tutorials-Practical) e.g. A course titled Operating System with course number TCT324 having 3 credits with 2 lectures, 1 practical and 2 tutorial be represented as TCT324(2-1-2).
The basic data for our problem is:
1. 12 courses of different credits (including 7 practical for 7 different courses)
2. 7 instructors to which the above courses have been assigned
3. 2 rooms where the lectures would be held

The detailed list of courses of Department of Computer Engineering, G.B. Pant University for a selected semester are described in the Tab 1.

Table 1. Data of Dept. of Computer Engineering, G.B. Pant University of Agriculture and Technology of a selected semester for generation of course timetable

S. No.	Abbrv.	Course Name	Credits	Year
1	DS	Data Structure	3(3-1-0)	II
2	DSS	Discrete Structure	3(1-0-0)	II
3	FLAT	Finite Lang. and Automata Theory	3(3-1-0)	II
4	FCCS	Fund. of Computer Comm. Systems	4(3-1-1)	II
5	PL	Programming Language	3(3-1-0)	II
6	SYP	System Programming	4(3-0-1)	III
7	DBMS	Database Management System	4(3-0-1)	III
8	ORS	Operational Research	2(2-1-0)	III
9	MIS	Management Information System	3(2-0-1)	III
10	CA	Computer Architecture	3(2-1-1)	IV
11	DMW	Data Mining and Warehousing	3(2-1-1)	IV
12	CGA	Computer Graphics and Animations	4(3-0-1)	IV

The hard constraints that are incorporated in the problem, as described in [6] are as:

1. A classroom must not be booked twice at one time.
2. Each event must be allotted exactly once.
3. Students must not have multiple classes simultaneously.
4. The classroom must have enough capacity to hold the allotted group of students.
5. Instructors must not be booked twice for one particular timeslot.
6. An instructor must not be allotted a lecture, laboratory or tutorial when he/she is unavailable. E.g. an instructor might have prior commitments.

The soft constraints, as described in [6] have been exempted from consideration since they vary greatly with respect to departments and universities.

4 Genetic Algorithm

Genetic Algorithms (GA) are adaptive systems inspired by natural evolution used for solving complex problems and for searching of large problem spaces. The power of GA lies in the fact that they are capable of finding the global optimum in multi-modal spaces [9]. This characteristic makes GA a well suited tool for timetabling problem.

4.1 Evolutionary Process

The Genetic Algorithm is derived from the process of evolution in nature, as stated by Charles Darwin [10]. This is done by generating a population of individuals represented as Chromosomes, in essence a set (an array) of character, binary or decimal strings [11]. The individuals in the population go through the process of evolution. Thus, producing new individuals in the population.
Initially, a fixed amount of random chromosomes (individuals) are generated equal to the population size. The fitness of these randomly generated individuals are evaluated using the defined constraint data. The individuals that are most fit i.e. having the highest maximum fitness value, are used for breeding to generate new chromosomes. This process is called Selective Breeding [6]. The chromosome produced after breeding is named as Generation. The new individual generated is again evaluated for its fitness value and if it has a better fitness value than the remaining in the population, the chromosome is added to the population and the one with least fitness value is discarded from consideration. The total number of individuals in a population (population size) remains constant throughout [10]. This process is repeated either depending upon the requirements or until a valid solution is obtained.

4.2 Genetic Algorithm Operators

4.2.1 Crossover

Once fitness of a chromosome is calculated, parents are chosen for breeding with maximum fitness value. A new creature is produced by selecting a part (a gene) of the chromosome from the first parent and another gene from the second parent. This process of combining genes is called Crossover. A crossover point is selected where the genes are formed from the chromosome. If one point is selected, it is called single-point crossover. Once the new chromosome has been produced it is added into the population [2] [12].

4.2.2 Mutation

After performing Crossover and before releasing child into the population there are chances that mutation may need to be performed. The probability of occurrence of mutation is called mutation rate [11]. Mutation is a process of injecting small genes or noise into the existing or newly generated chromosome. The need of mutation arises when the chromosome produced is very unfit or has the solution that is far from required. Thus, a noise is injected into the chromosome, for increasing its chances to be included in the chromosome [2] [5] [13].

5 Proposed Methodology

Initially, a colony is established using random variables with colony size equal to population size. The fitness value of these randomly generated individuals (chromosomes) are calculated and sorted accordingly. Since, the colony size is equal to the population size no offspring is discarded (killed) at this time.
If the randomly generated individuals have fitness value of 1.0, the happiness value is calculated (one at a time) for such chromosomes directly. This happiness value is initially stored in the memory. If there is more than one individual with fitness value of 1.0, the happiness value is calculated for all such individuals. The initially stored happiness value in the memory is overridden, if at later stages the individuals are found to have happiness value greater than the value stored previously.
If the randomly generated individuals do not have fitness value of 1.0, they are bred to generate offspring. The breeding is performed on those individuals (parents) which have maximum fitness value. The two parents bred together to

generate an offspring. Again, the fitness value of the newly generated offspring is calculated. If the newly generated offspring has fitness value greater than the minimum value of fitness in the colony, the newly generated offspring is added into the colony and the offspring with the minimum fitness is discarded from the colony. This process of breeding continues until offspring with fitness value of 1.0 is not obtained.
The Happiness value is calculated for the new individuals in the colony with fitness value of 1.0. And again, if the happiness value is found to be greater than the value stored in the memory; the old value is overridden with new happiness value. This entire process is repeated until total generations are produced. The flow chart for genetic algorithm along with happiness parameter is shown as a schematic diagram in Fig. 1.

Before calculating fitness value, the timetables are checked for any error that might occur as a result of random timetable generation or due to breeding. A concept of repair strategy is used to produce repaired timetables. This is a part of hybrid Genetic Algorithm, as suggested by Gen and Cheng [14]
Some genes of the chromosomes are outside the search space. This might include for example, two timetable bred to form a new timetable and in the resulting timetable one class has more than one instance [2] [6].
In this paper, there are two stages of repair strategy used for repairing errors caused by breeding of timetables as proposed by [6]. Firstly, a class may be booked more than once in a timetable. The total number of one type of classes booked are calculated and if more than one instance of the class is found, a randomly chosen class is altered as null booking. Secondly, a class may not be booked at all in the resulting timetable. Thus, all rooms are searched for that class and if no room is found with that class, a null booking is taken randomly and class is allotted to that null booking.

5.1 Happiness Parameter

Before calculating fitness value, the timetables are checked for any error that might occur as a result of random timetable generation or due to breeding. A concept of repair strategy is used to produce repaired timetables. This is a part of hybrid Genetic Algorithm, as suggested by Gen and Cheng [14]. 5.1 Repair Strategy
The happiness parameter calculates happiness of the entire timetable. The happiness parameter is calculated after implementation of GA on those individuals that have fitness value of 1.0.
The Happiness parameter is the sum of number of free time slots an instructor has after which no lectures are scheduled for that particular day, for all days of the week. A penalty (-1) is added for every lecture scheduled early morning 08:00-09:00 and/or immediately after lunch (13:00-14:00) from 14:00-15:00. The algorithm to calculate happiness parameter is shown in Algorithm 1.

The formula to calculate the Happiness value is:

$$Happiness = \sum_{i=0}^{n} \{(Total\ Time\ slots(i.e.8) - Position\ of\ last\ lecture\ for\ i\ day) - p\}$$

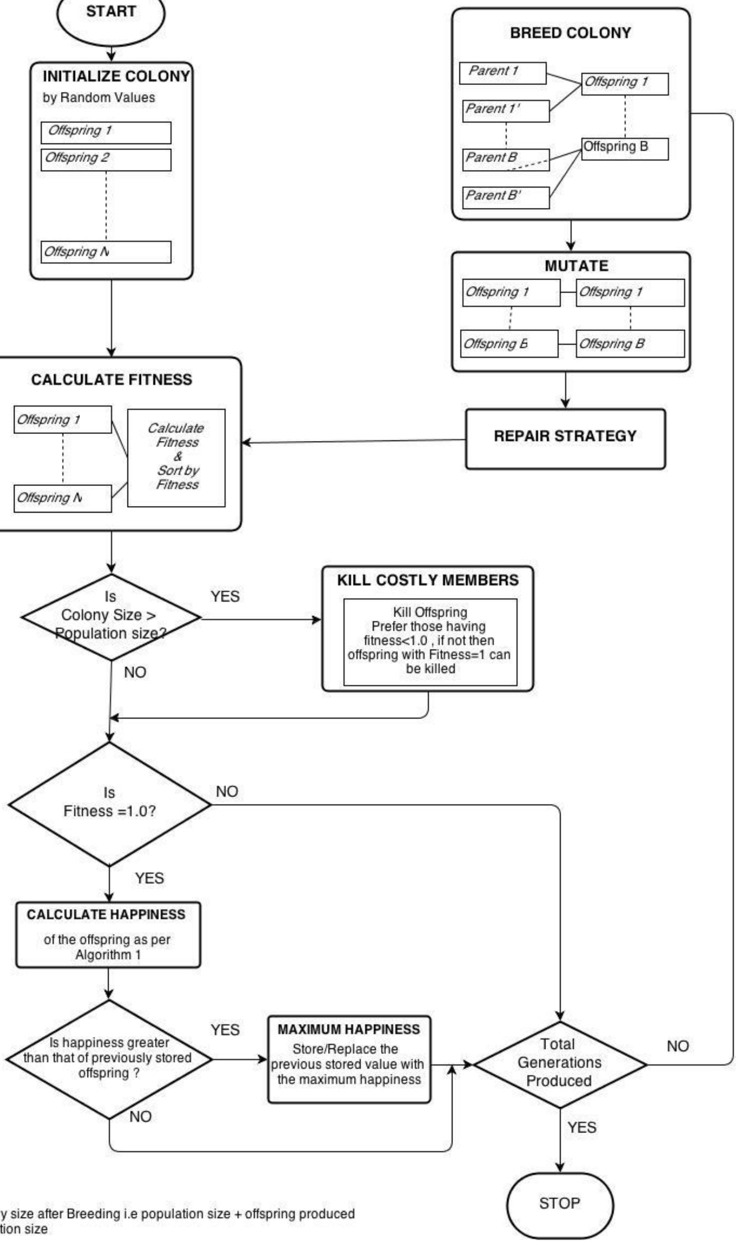
Where, i = Number of Working days {Mon, Tue, Wed, Thu, Fri, Sat}, and

$$\mathbf{Penalty\ , p} = \begin{cases} -1, & \boldsymbol{If\ a\ lecture\ is\ scheduled\ at\ 08{:}00-09{:}00\ OR\ 14{:}00-15{:}00} \\ -2, & \boldsymbol{If\ a\ lecture\ is\ scheduled\ at\ 08{:}00-09{:}00\ AND\ 14{:}00-15{:}00} \end{cases}$$

Algorithm 1: Algorithm for calculating Happiness Parameter

Input: Timetable
Output: Happiness parameter value

```
var day = {Mon, Tue, Wed, Thu, Fri, Sat}
var timeslot = {08:00-09:00, 09:00-10:00, 10:00-11:00,
11:00-12:00, 12:00-13:00, 13:00-14:00, 14:00-15:00,
15:00-16:00, 16:00-17:00}
var Happiness = 0;
var free_timeslots = 0;
var last_allocated_timeslot = 0;
for days = 0 to 5 do
  Calculate last_allocated_timeslot;
  Happiness = Happiness + free_timeslots;
  for timeslot = 0 to 8 do
      if Lecture allocated at timeslot = 08:00-09:00
         OR 14:00-15:00
      then
            Happiness = Happiness - 1;
      end
      else If Lecture allocated at timeslot =
               08:00-09:00 AND 14:00-15:00
      then
            Happiness = Happiness - 2;
      end
  end
  return Happiness;
  end
```


START
INITIALIZE COLONY
by Random Values
Offspring 1
Offspring 2
Offspring N
CALCULATE FITNESS
Offspring 1
Offspring N
Calculate Fitness & Sort by Fitness
BREED COLONY
Parent 1
Parent 1'
Offspring 1
Parent B
Parent B'
Offspring B
MUTATE
Offspring 1
Offspring 1
Offspring B
Offspring B
REPAIR STRATEGY
Is Colony Size > Population size?
YES
NO
KILL COSTLY MEMBERS
Kill Offspring Prefer those having fitness<1.0 , if not then offspring with Fitness=1 can be killed
Is Fitness =1.0?
NO
YES
CALCULATE HAPPINESS
of the offspring as per Algorithm 1
Is happiness greater than that of previously stored offspring ?
YES
NO
MAXIMUM HAPPINESS
Store/Replace the previous stored value with the maximum happiness
Total Generations Produced
NO
YES
STOP

where, B =colony size after Breeding i.e population size + offspring produced
N=population size

Figure. 1. Flowchart for generating course timetable using GA and Happiness Parameter

6 Result and Discussion

The proposed algorithm was programmed using C++ and Java (for GUI) and simulation was performed on Intel core i3 processor 2.4GHz running Windows Operating system and tested on the data of Department of Computer Engineering, G.B. Pant University of Agriculture and Technology. The specification considered for the GA are mentioned in Tab. 2. The Tab. 3 shows the result of simulations obtained by executing the algorithm on the same data set. The number of generations with maximum and minimum Happiness are also specified in the table.

The generations with fitness value of 1.0 are relatively very large in number and the solution with maximum happiness are very small in number. There are many generations in which the Happiness is negative and the minimum Happiness value has been specified in the table.

Table 2. Specification for Genetic Algorithm

S. No.	Parameters	Quantity
1	Number of Generations	20,000
2	Population Size	50
3	Number of Working Days	6
4	Number of Timeslots each day	8

The graphs with varying fitness and happiness values depicting the simulation result are shown in Fig. 2 and 3. Fig. 2 shows the fitness value on the Y axis and number of generations on the X-axis. Similarly, Fig. 3 show the Happiness parameter value on the Y axis and number of generations on the X-axis. The graph clearly states that the number of generations with positive happiness are very less and the generations with negative happiness are greater in number. The average Happiness was calculated for the simulations and was found to be negative. In Fig. 3 we have shown the best possible curve of Happiness factor.

The probability that the selected solution is having negative happiness is calculated using following formula.

$$Probability\ of\ selecting\ solution\ with\ negative\ Happiness$$
$$= \frac{Total\ number\ of\ Generations\ with\ negative\ Happiness}{Total\ Number\ of\ Generations\ having\ Fitness}$$

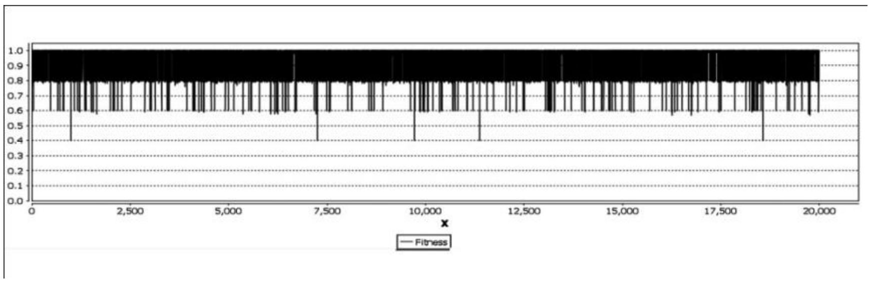


Figure. 2. Simulation Graph with Fitness=1.0

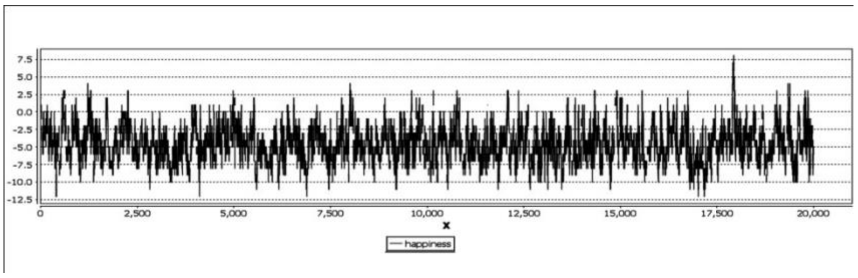


Figure. 2. Simulation Graph with maximum Happiness=8.0

Table 3. Simulation Results on Application of Algorithm 1

S. No.	G_0	$G(H_{neg})$	H_{max}	$G(H_{max})$	H_{min}	$G(H_{min})$	H_{avg}
1	14288	13566	5	2	-12	19	-4.608133
2	14317	13488	6	4	-13	6	-4.650268
3	14228	13300	6	1	-14	1	-4.556508
4	14190	13476	6	8	-12	5	-4.834179
5	14254	13514	6	11	-13	4	-4.687877
6	14427	13537	6	10	-13	17	-4.635198
7	14300	13192	7	2	-13	2	-4.419161
8	14422	13461	8	1	-12	6	-4.498613
9	14290	13298	5	9	-12	14	-4.348775
10	14158	12842	6	15	-13	3	-4.323138

Where, G_0 = Generations with Fitness = 1.0,
$G(H_{max})$ = Generations with Maximum Happiness,
$G(H_{min})$ = Generations with Minimum Happiness
$G(H_{neg})$ = Generations with negative Happiness,
H_{max} = Maximum Happiness
H_{min} = Minimum Happiness
H_{avg} = Average Happiness

The Tab. 4 represents the probability of selection of a timetable with negative happiness for all the iterations done in Tab. 3. The values in Tab. 4 clearly depicts that there are more than 90% (approx.) chances of selection of non-appropriate timetable without using Happiness parameter. Hence, using Happiness parameter the probability of selection of an appropriate feasible solution is greatly increased and that of non-appropriate solution reduces gradually.

Table 4. Probability of selecting solutions with negative Happiness

S. No.	**G_0**	$G(H_{neg})$	**Probability of selecting offspring with** H_{neg}
1	14288	13566	0.94946
2	14317	13488	0.93855
3	14228	13300	0.93477
4	14190	13476	0.94968
5	14254	13514	0.94808
6	14427	13537	0.93831
7	14300	13192	0.92251
8	14422	13461	0.93336
9	14290	13298	0.93058
10	14158	12842	0.90704

We believe that adding this happiness parameter in the genetic algorithm will narrow the solution space and would generate timetables that will meet the comfort of the instructors and the students. The solutions with less happiness would be discarded from consideration of final solution.

7 Conclusion

This paper introduces a Happiness parameter with Genetic Algorithm. We have shown that using Happiness parameter, an appropriate and feasible timetable solution is obtained. Depending upon the conditions the non-appropriate solutions are discarded. Future work will be aimed to enhance the algorithm by providing Happiness for individual Instructors. We will also aim for making this parameter versatile so that it can be used for other problems like motorcycle assembling in the field of production.

References

[1] Tim B. Cooper and Jeffrey H. Kingston. The complexity of timetable construction problems. In Edmund Burke and Peter Ross, editors, *Practice and Theory of*

Automated Timetabling, volume 1153 of *Lecture Notes in Computer Science,* pages 281-295. Springer Berlin Heidelberg, 1996.

[2] Manar Hosny and Shameem Fatima. A survey of genetic algorithms for the university timetabling problem. *International Proceedings of Computer Science and Information Technology,* vol. 13, 2011.

[3] David Come and Peter Ross. Peckish initialisation strategies for evolutionary timetabling. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling,* pages 227-240, London, UK, UK, 1996. Springer-Verlag.

[4] M.F. Zibran. *A Multi-phase Approach to University Course Timetabling.* Canadian theses. University of Lethbridge (Canada), 2007.

[5] Rhydian Lewis and Ben Paechter. Finding feasible timetables using group-based operators. *IEEE Trans. Evolutionary Computation,* 11(3):397-413, 2007.

[6] Leon Bambrick. Lecture timetabling using genetic algorithms. *The University of Queensland,* 1997.

[7] S. Abdullah and H. Turabieh. Generating university course timetable using genetic algorithms and local search. In *Convergence and Hybrid Information Technology, 2008. ICCIT '08. Third International Conference on,* volume 1, pages 254-260, 2008.

[8] Shengxiang Yang and S.N. Jat. Genetic algorithms with guided and local search strategies for university course timetabling. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* 41(1):93-106, 2011.

[9] Kurt A Hacker, John Eddy, and Kemper E Lewis. Efficient global optimization using hybrid genetic algorithms In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* pages 4-6, 2002.

[10] J.H. Holland. *Adaptation in natural and artificial systems:* an *introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, 1975.

[11] L. Davis. *Handbook of genetic algorithms.* VNR computer library. Van Nostrand Reinhold, 1991.

[12] Salwani Abdullah, Hamza Turabieh, Barry McCollum, and Edmund K Burke. An investigation of a genetic algorithm and sequential local search approach for curriculum-based course timetabling problems. In *Proc. Multildisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009), Dublin, Ireland,* pages 727-731, 2009.

[13] V. Sapru, K. Reddy, and B. Sivaselvan. Time table scheduling using genetic algorithms employing guided mutation. In *Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference on,* pages 1-4, 2010.

[14] M. Gen and R. Cheng. *Genetic algorithms and engineering design.* Wiley series in engineering design and automation. Wiley, 1997