Class Timetable Scheduling with Genetic Algorithm

Article · December 2013

CITATIONS
0

READS
5,587

2 authors:

Rajaram Ambole
Vidya Pratishthans Kamalnayan Bajaj Institute of Engineering and Technology
2 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Dinesh Bhagwan Hanchate
Vidya Pratishthan's, College of Engineering, Baramati
106 PUBLICATIONS **92** CITATIONS
SEE PROFILE

Class Timetable Scheduling with Genetic Algorithm

[1]Rajaram H. Ambole, [2]Dinesh B. Hanchate
[1,2]Dept. of Computer Engineering, Vidya Pratishthan's College of Enggineering, India

Abstract
Genetic Algorithm is population based heuristic method extensively used in scheduling applied for constraint optimization problem. In Genetic Algorithm we generated random solutions rst called as populations and then try to generate feasible solution with the help of operations like selection, crossover and mutation. Scheduling class timetable is common scheduling problem in which a set of events is to be arranged in available timeslots along with limited resources and constraints. Selection of solutions is an important function in genetic algorithm which decides the quality of ospring generated. Selection is a way to make space for good solution to in and bad solution to out from population. In this project, we worked with small instance of timetable problem which required scheduling 100 events in 45 timeslots. We used tournament selection II and tournament selection V to check the quality of ospring generated. We found considerable improvement in solution generated with tournament selection V.

Keywords
Genetic Algorithm, Hard and Soft Constraint, Timetable, Local Search

I. Introduction

CLASS Timetable is an important activity for any of educational institutes from the administration point of view. It is resource and timeline constraint problem. Class timetable concerns all activities with regard to making a timetable. According to Collins Concise dictionary (4th Edition)" a timetable is a table of events arranged according to time when they take place." The events are actually meetings between people at particular location. A timetable must meet requirement of people involved in it with possibilities. Actually timetable is common scheduling problem which can be described allocation of resources for the specic task under predened constraints [1].

Genetic Algorithm is population based heuristic method largely applied to solve scheduling problems. It is a search method based on principles of natural selection and genetics [2]. It encodes decision variable of search problem into nite length of string alphabets of certain cardinality. Shengxiang Yang and Sadaf Jat used local and guided search in genetic algorithm to get feasible solution with minimum soft constraint violation [3]. Rhydian Lewis and Ben Paechter used an impirical analysis and shown improved results for timetable [4]. Salwani Abdullah and Hamza Turabeih proposed local search method to solve class timetable [5]. R Sivaraj and Dr. T. Ravichandran had shown eective analysis regarding the selection methods available for genetic algorithm. They also had shown how selection is used to improve the chance of survival of ttest individual [6].

The paper is organized as follows:

- Section II Problem Description.
- Section III Problem Formulation.
- Section IV Implementation.
- Section V Experimental Results.
- Section VI Conclusion.

II. Problem Description

Timetabling problem considered here is actually reduction of typical class timetabling problem. According to Rossi-Doria [1] the problem consists of a set of events of classes E to be scheduled in available timeslots along with a set of rooms R in which events can take place and the set of students S who can attend these events and set of fea- ture F which is required by events and satised by rooms. A feasible timetable is one in which all events have been assigned to room and timeslot so that following hard constraints are satised.

- No student can attend more than one event at the same time.
- The room is big enough to schedule the event and it must satisfy features required by the event.
- Only one event can be scheduled in a room at any timeslot.

In addition to this timetable is equally penalized for the occurrences of following soft constraints.

- Students attend class in the last slot of the day.
- Students attend more than two classes in row.
- Students attend single class on a day.

Note that the soft constraint selected is representative of three dierent classes; rst one checked with no knowledge of rest of timetable; second one checked while building a solution taking into account that events are assigned to nearby timeslots and nally last one checked only when timetable is complete and all events are assigned to timeslots. The main objective here is to minimize number of soft constraint violation while generating feasible solution. A solution is feasible if it does not have any hard constraint violation.

III. Problem Formulation

The problem is to schedule given number of events. We are attempting to nd the satisfactory choice when and where to schedule the events for given number of students, number of rooms and features available in each room. An instance of this scheduling problem consists of scheduling set of n events (classes and subjects) E ={e1, e2, . . . , en} to be scheduled into a set of 45 time slots T ={t1, t2, . . . , t45} (i.e., nine for each day in a ve days week), a set of m available rooms R = {r1, r2, . . . , rm} in which events can take place, a set of k students S = {s1, s2, . . . , sk} who attend the events, and a set of l available features F = {f1, f2, . . . , fl} that are satised by rooms and required by events.

The schedule is then mapping a room along with timeslot which we can express as follows.

f(a,b) | a ∈ {t1, t2, . . , t45}, b∈ {r1, r2, . . . , rm} }

Interrelationships between these sets are given by ve matrices. The rst matrix A[k,n], called Student-Event matrix, shows which event is attended by which students. The second matrix B[n,n] called Event-Con ict matrix indicates whether two events can be scheduled in the same timeslot or not. It helps to quickly identify events that can be potentially assigned to the same timeslot. The third matrix C[m,l] called Room-Features matrix gives the features that each room possesses. The fourth matrix D[n,l] called Event-Features matrix gives the features required by each event. The last matrix G[n,m] called Event-Room matrix lists the possible rooms to which each event can be assigned. Through this matrix

we can quickly identify all rooms that are suitable in size and feature for each event. Usually, a matrix is used for assigning each event to a room ri and a time slot ti . Each pair of (ti , ri) is assigned a particular number that corresponds to an event. If a room ri in a time slot ti is free or no event is placed, then "|1" is assigned to that pair.

The generated solution quality is calculated based upon number of soft constraint violations. The solution having least soft constraint violation is called feasible solution. For this we used following objective function [3].

f(s) := HCV(s) * C +SCV(s) (1)

Where C is a constant, which is larger than the maximum possible number of soft-constraint violations.

IV. Implementation

A. Genetic Algorithm

We used Genetic Algorithm to solve class scheduling problem with custom selection methods. Genetic algorithm is population based heuristic method largely applied to solve the scheduling problems. It is search methods based on principles of natural selection and genetics. The basic working of genetic algorithm is given by following algorithm.

As stated algorithm Genetic Algorithm starts with initializing the population with some random solutions. After initialization we evaluated these random solutions to determine the survival capacity of solution. Selection is an important activity used here to select proper parents to generate an individual called ospring which improves probability of survival of good solution. We applied crossover and mutation with probability [0.-0.8] to generate ospring from selected solutions so called parents and then again evaluate this ospring for its quality. If this ospring is better than available individuals in population then will replace this ospring into population otherwise discarded. We are repeating this process till termination condition is reached. Termination condition is total number of times you want to execute this loop activity.

Algorithm 1: Basic Genetic Algorithm

Input: Problem Instance
Output: A solution to problem instance

```
1  Initialization of Population
2  Evaluate each individual from Population
3  Set Generations
4  i ← 0
5  while NOT termination or i ≤ Generations do
6      Select Parents
7      Recombine Pairs of Parent to generate child
8      Evaluate Child
9      Update Population
10     i ← i + 1
11 return Solution
```

B. Chromosome Representation

We represented chromosome as ordered pair of decimal number. The rst decimal will represent timeslot and second decimal will represent classroom number while pair itself indicates event number.

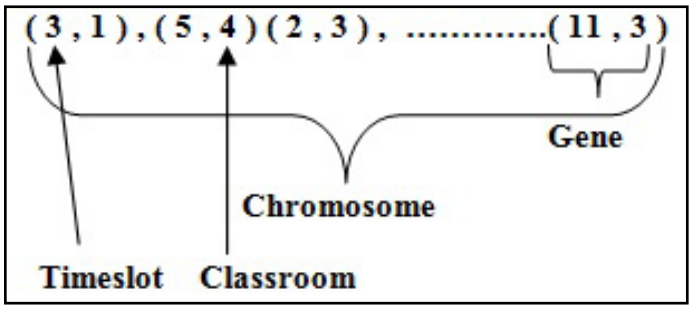

Fig. 1: Chromosome Representation

C. Initializing Population

We initialized with 10 individuals to keep population size 10. In initializing the population we simply assigned a timeslot on random basis to all events and then assigning possible rooms to each event by considering the requirements of each events. For generating random timeslot we used random number in the range [0-1],

Timeslot = (integer) (random (0, 1) * 45)

D. Evaluation of Every Individual

Evaluation is an important step to carry before passing solutions to Genetic Algorithm. In evaluation we evaluated tness of each individual. We evaluated population by applying local search algorithm to every individual present in population. Local search algorithm [3] basically tries to improve the individual by removing Hard Constraint Violations (HCV) and then by removing Soft Constraint Violations (SCV). It used three dierent moves called N1, N2 and N3 discussed in section G.

We used three dierent moves to remove hard and soft constraint from every individual eectively. First random

Algorithm 2: Local Search Algorithm

Input: Population $\langle E_1, E_2, \ldots, E_n \rangle$
Output: Improved Individual

```
1  ∀ e ∈ E
2  If ∑_i^n HCV_E ≥1
3  Apply N1 Move, if N1 fails then
4  Apply N2 Move, if N2 fails then
5  Apply N3 Move to remove Hard Constraints
6  End if
7  ∀ e ∈ E
8  If ∑_i^n HCV_E =0
9          If ∑_i^n SCV_E ≥1
10         Apply N1 Move, if N1 fails then
11         Apply N2 Move, if N2 fails then
12         Apply N3 Move to remove Soft Constraints
13         End if
14 End if
15 return Improved Individual
```

move is used to assign an event to any random timeslot. Second random move is used to swap the timeslots of two dierent events in an individual while Third random move is used to permute any three events in an individual and produce some dierent combination than the available. Then use this combination is used to improve the tness of an individual. In above algorithm lines 1 to 6 are used to remove HCV and lines 7 to 13 are used to remove SCV (if possible) from an individual. After applying local search to all individuals in population, it is sorted out according to their tness values.

E. Selection of Parents

Our focus is to have correct selection of individual called parents who are used to generate ospring (child). If selected parents are t and capable to survive then the generated child may have same capability to survive. In this approach we used 2 selection methods.

1. Tournament Selection II
2. Tournament Selection V

Algorithm 3: Tournament 2 Selection

Input: Population $\langle E_1, E_2, \ldots, E_n \rangle$
Output: Single Individual as Parent

```
random1 = random(0,1)
Parent1 = random1 * Size_population
random2 = random(0,1)
Parent2 = random2 * Size_population
if Parent1_fitness ≥ Parent2_fitness
return Parent1
else
return Parent2
```

First method considers part of solution where second method considers proportion of solution. In tournament selection II have selected individuals randomly by using following technique and then selected one of them as parent with the help of tness associated with individualand again applied the same technique to select second parent.

Algorithm 4: Tournament 5 Selection

Input: Population $\langle E_1, E_2, \ldots, E_n \rangle$
Output: Single Individual as Parent

```
random1 = random(0,1)
BestParent = random1 * Size_population
for i ← 1 to 5 do
    random2 = random(0,1)
    Parent = random2 * Size_population
    if Parent_fitness > BestParent_fitness
    BestParent ← Parent
return BestParent
```

F. Crossover

We used custom method to generate ospring from the selected parents using tournament selection. Crossover probability is set to 0.8. Actually we have implemented multipoint crossover and these multi points are decided by random number we have generated in a range 0 to 1. In crossover we selected some timeslots from parent1 and some timeslots from parent2 on random basis. After selecting timeslots for all events we assigned rooms to each event as per its requirement.

Parent 1 (1 , 3) , (2 , 3) , (3 , 4) .....(2, 4)
Parent 2 (2 , 3) , (4 , 3) , (23, 1).....(12, 4)
Child (1 , ?) , (4 , ?), (3 , ?).....(2 , ?)

In the above solution, timeslot selection is done on the basis of random number generated in the range of 0 to 1. After selecting timeslots we assigned rooms to each events as per its requirement.

Algorithm 5: Crossover

Input: Parent1 and Parent2
Output: Child

```
∀ e_i ∈ E
for i ← 0 to n do
    random1 = random(0,1)
    if (random1 < 0.5)
    Child_timeslot ← Parent1_timeslot
    else
    Child_timeslot ← Parent2_timeslot
    End if
return timeslot
```

G. Mutation

Child generated in crossover has undergone through mutation based on probability < 0.5. Mutation is a technique which actually changes characteristics of single individual. Here the single individual consists of ordered pair of timeslot and classrooms. In this we applied techniques used in local search.

Child (2 , 3) , (4 , 3) , (23, 1)(12, 4)
If N1: (5 , 3) , (4 , 3) , (23, 1)(12, 4)

By changing timeslot associated with any random event (Here rst event selected)

Child (2 , 3) , (4 , 3) , (23, 1)(12, 4)
If N2: (4 , 3) , (2 , 3) , (23, 1)(12, 4)

By swapping timeslots with two successive events (Here rst two events are selected.

Child (2 , 3) , (4 , 3) , (23, 1)(12, 4)
If N3: (23, 4) , (3 , 4) , (1, 3)..(12, 4)

By taking three successive events and permuting them to generate dierent events than available. (Here rst three are used for permutation)

Algorithm 6: Mutation

Input: Child Generated in Croosover
Output: Mutated Child

```
random1 = random(1,3)
if (random1 = 1)
Apply N1 Move
else if (random1 = 2)
Apply N2 Move
else
Apply N3 Move
End if
return Child
```

H. Ospring Evaluation

Evaluation is actually testing the tness of a generated ospring using crossover and mutations. This generated ospring underwent through local search to improve its tness. This improved ospring is then compared with available solutions. If ospring is better than any of individuals present in population then we put it in population by replacing less improved individual exist in population.

I. Termination Condition

We use total generations to terminate algorithmand if the total number of generations reached to the required level; stop the

algorithm execution else repeat all steps.

V. Experimental Results

We have carried out timetable generation on the small instance of problem solution available. There are total three dierent instances of problems which are available. We have tested total 5 small instances for making the schedule. These 5 small instances have characteristics which are given in table 1. The application was coded in C++ and tested with Visual Studio 2010 professional Edition. The obtained results are slightly diering as we changed the selection method. All tests are carried out on Intel Core i3 CPU with 2.27 GHz Processor and 3 GB RAM. The environment is Windows 7 Operating system. We used standard benchmarked data set available with WATT -The EURO Working group on Automated TimeTabling. We found dierence in the tness of nal result if the selection method is changed.

Table 1: Avaialbe Problem Instance

Problem Instance	Small	Medium	Large
events	100	400	400
rooms	5	10	10
Features	5	5	10
students	80	200	400
Features per room	3	3	5
Max. Students per event	20	20	20
Max. Eevents per student	20	50	10

A. Results

Results were observed with set of 5 small problem instance having dierence in class room size and maximum students per events. Following parameters are xed for the execution.
Population Size = 10, Crossover Probability < 0.8
Mutation Probability < 0.5,Number of Generations = 50

Table 2 : With and Without Local Search(LS)

Instance	SFitness without LS	Fitness With LS
1	Infeasible	06
2	Infeasible	14
3	Infeasible	13
4	Infeasible	08
5	Infeasible	06

By above table it is clear that only crossover and mutation are not sucient to generate a solution. We must have tness function deciding life of particular solution.

Table 3: With Tournament Selection 2 and 5

Instance	Fitness With TS2	Fitness With TS5
1	06	5
2	14	13
3	13	13
4	08	08
5	06	05

Above results show considerable change when we applied tournament selection5 instead of tournament selection2. It shows that considering the proportion of available solutions is better that considering the part of solution. We observed better tness with tournament 5 selection.

Table 4: With Increasing Generation and constant Population

Instance	Generation	SCV
1	10	06
1	15	06
1	20	03
1	25	03
1	30	03
1	35	02
1	40	02
1	45	02
1	50	02

Table 5: With Increasing Population and constant Generation

Instance	Population	SCV
1	10	06
1	15	06
1	20	06
1	25	06
1	30	06
1	35	06
1	40	06
1	45	06
1	50	06

B. Graphs

From the below Graph , It is prooved that considering the proportion of the solution space instead of the part shows signicant dierence in generated output. In tournament selection 2, to select parents we compared only two random individuals from population Whereas in Tournament selection 5 we compared ve individuals from population. Probability of selecting t individual as a parent is increased in Tournament Selection 5.

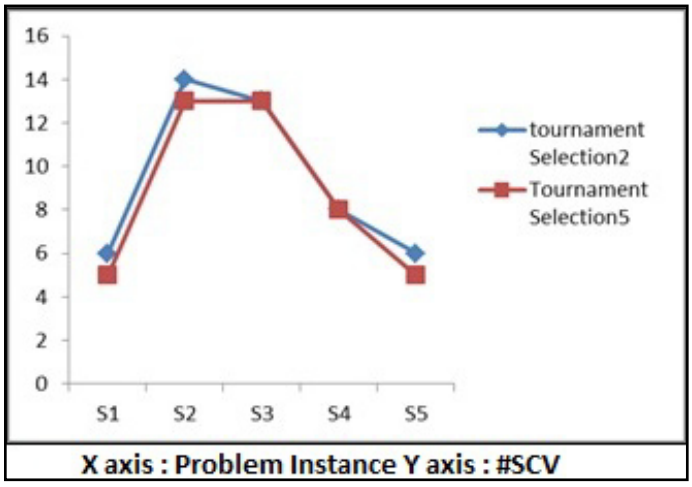

Fig. 2: Tournament Selection2 Vs Tournament Selection 5

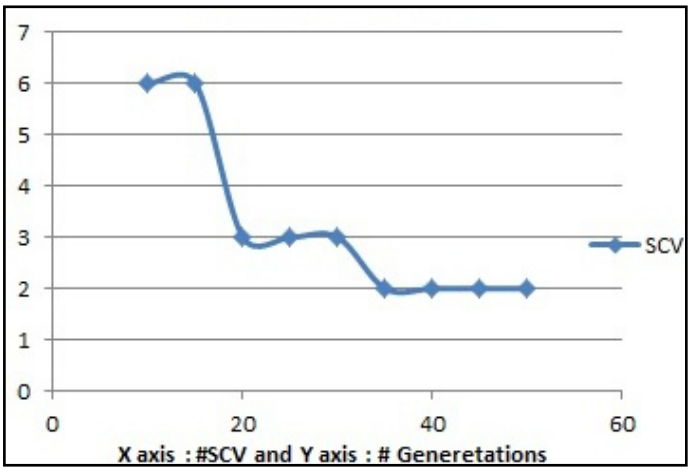

Fig. 3: Generation Vs Fitness

VI. Conclusion

Tournament Selection 2 is considering only two individuals from entire population and therefore sometimes we actually miss to select good parents. If we apply tournament selection 5 the chance to select good parent is increased and thus the chance to generate child ospring with good quality also gets increased. Therefore selection is an important aspect of Genetic Algorithm which can have strong impact on it's performance.

On other hand, the local search plays a vital role while generating a solution for a timetable. Actually it is a core part of Algorithm that deals with hard constraint removal and soft constraint removal. Sometimes we are getting feasible solution just by applying local search. Therefore by applying local search eectively for each individual one can reduce the total number of generations required to produce feasible solution.

Number of generation is one more important factor than the population size which makes signicant dierence to the schedule.

References

[1] O. Rossi-Doria, M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, T. Stutzle,"A comparison of the performance of dierent metaheuristics on the timetabling prob-lem, Proc. 4th Int. Conf. Pract.Theory Automated Timetabling (Lecture Notes in Computer Science), vol. 2740, 2003, pp. 329-351.

[2] K. Sastry, D. Goldberg, G. Kendall, Genetic algorithms, Genetic algorithms, Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques, E. K. Burke and G. Kendall Eds. New York: Springer- Verlag, 2005, ch. 4, pp. 97-125

[3] S. N. Jat, S.Yang A guided search genetic algorithm for the university course timetabling problem, 4th Multidisciplinary Int. Conf.Scheduling: Theory Appl., 2009, pp. 180-191.

[4] R. Lewis,"A survey of metaheuristic based techniques for university timetabling problems, OR Spectrum, Vol. 30, No. 1, pp. 167-190, 2008.

[5] Salwani Abdullah, Hamza Turabieh,"A Hybrid evolutionary approach to the University course timetabling problem, 2007 IEEE Congress on Evolutionary Computation (CEC 2007).

[6] S. Sivaraj, Dr. T. Ravichandran A Review of Selection methods in Genetic Algorithm", International Journal of Engineering Science and Technology 2011.

Rajaram Hanumant Ambole Received B.E. degree in Computer Engineering from Vidya Pratishthan0s College of Engineering (VPCOE), Baramati and pursuing his M.E. degree in Computer Engineering from the same institute i.e. VPCOE Baramati (Pune University). He has 9 years of teaching experience. His areas of interest are Genetic Algorithm, Project Scheduling and Compiler.

Dinesh Bhagwan Hanchate Birth Place :- Solapur, B.E. Computer from Walchand College of Engineering, Sangli (1995), Lecturer in Gangamai College of Engineering,Dhule (1995-96), Lecturer in S.S.V.P.S.s B.S.D. College of Engineering,Dhule in Computer and IT depts. (1996-2005), M.Tech.in Computer from Dr. Babasaheb Ambedkar Technological University, Lonere(2002-05), Currently as Asst. Prof. Computer Engineering, former H.O.D. (Computer and IT) in Vidya pratishthan0s College of Engineering, Baramati. Currently doing research (SGGS0s Institute of Technology and Engg, Nanded aliated to SRTMU,Nanded) under the guidance of Dr. Bichkar R.S., Dean, Research at G.H. Raisonis College of Engineering and Management,Wagholi, Pune.