HIGH SCHOOL TIMETABLING USING TABU SEARCH AND PARTIAL FEASIBILITY PRESERVING GENETIC ALGORITHM

Sanjay R. Sutar[1] and Rajan S. Bichkar[2]
[1]Asso. Prof., Dr. B. A. T. University, Lonere, Research Scholar, SGGSIET, Nanded, India.
[2]Professor, E&TC and Dean R&D, G. H. Raisoni College of Engg. & Mgt., Pune, India.

ABSTRACT

The high school timetabling is a combinatorial optimization problem. It is proved to be NP-hard and has several hard and soft constraints. A given set of events, class-teacher meetings and resources are assigned to the limited space and time under hard constraints which are strictly followed and soft constraints which are satisfied as far as possible. The feasibility of timetable is determined by hard constraints and the soft constraints determine its quality. Difficult combinatorial optimization problems are frequently solved using Genetic Algorithm (GA). We propose Partial Feasibility Preserving Genetic Algorithm (PFP-GA) combined with tabu search to solve hdtt4, "hard timetabling" problem a test data set in OR-Library. The solution to this problem is zero clashes and maintaining teacher's workload on each class in given venue. The modified GA procedures are written for intelligent operators and repair. The PFP-GA in association with Tabu Search (TS) converges faster and gives solution within a ***few*** *seconds. The results are compared to that of using different methodologies on same data set.*

KEYWORDS: *Genetic Algorithm, Hard Timetabling, Intelligent Operators, Repair*

I. INTRODUCTION

The timetabling is one of the major activities for an institution. It is an optimization problem and supposed to be a subset of scheduling problems. Timetabling is a problem of assigning a number of events into a number of time slots. The problem has many forms like educational timetabling, employee timetabling, sports timetabling, transportation scheduling, etc. Timetabling problems are computationally complex, constrained optimization problems. Main objective of such constraint satisfaction problem is to assure all constraints, rather than optimizing a number of objectives. Automated timetabling saves a lot of man-hours and provides optimal solutions within short time which can boost productivity, quality of education and services.

The NP hard problems are computationally complex hence computations in order to explore an optimal result increase exponentially with the structure of the problem. The goal of timetabling process is satisfying certain set of constraints with respect to variables like rooms, teachers, time slots, classes, courses etc. Institutions will have different protocols for maintaining the timetables for the students. Therefore, it is difficult to design a common framework which will work for all the instances. Timetabling problem may not be completely automated as one solution may be better than other, also as the search space is large, manual technique may lead to better optimization.

Educational institutions have to perform this task regularly which requires a large time and efforts. Hence designing techniques for the automated timetables is still of interest. The school timetabling problem basically involves assigning class, teacher and room tuples to time slots, a required number of times. The hard constraints include all class-teacher interactions must be scheduled, no class or teacher must be scheduled more than once in a slot, certain classes may need to be split and regrouped for certain lessons. Timetable will still be feasible though the soft constraints are not satisfied.

Minimizing the soft constraints cost improves the quality of the timetable. Typical soft constraints include class or teacher preferences.
A large number of methods for solving timetabling problems come from disciplines like Evolutionary Techniques and Computational Intelligence, Artificial Intelligence, Fuzzy Logic, Operations Research, Local Search, and development in solving them is a major objective of research in these areas. The metaheuristics techniques seem to be efficient approach for solving hard combinatorial problem as they can generate solutions in reasonable time. Also they can be applied to different variants of problems.

Genetic Algorithm is population based technique which can be applied to resolve any problem. It is really a powerful optimization tool. GAs are used to solve optimization problems similar to the evolution. Generation phase creates random individuals to constitute an initial population. Each individual represents one of the possible solutions and the population is then modified in an evolutional process. The algorithm terminates when stopping criteria is met. GAs produce timetables comparable to that of used by the researchers.

Tabu Search (TS) is an iterative technique for solving combinatorial optimization problems which uses memory structures as guide to continue exploration. The TS was independently proposed by Glover and Hansen. It Starts with an initial solution s, explores the neighborhood N(s) and selects the best admissible movement m, such that the application of m in the solution s produces the new solution. A mechanism called short-term memory is used to avoid repetition. Its objective is to restrict movements towards already visited solutions. The movements are stored in a tabu list with tabu status, for a given number of iterations, called tabu tenure. The movements with tabu status are accepted to allow high quality solutions, if the cost of the new solution produced satisfies an aspiration criterion. Promising regions of the search space are investigated with intensification and each region of the search space is explored with diversification. The main contribution of this paper is a combination of PFP-GA and tabu search that provides a faster solution to hdtt problem.

The following section provides previous work employing tabu search to solve the timetabling problem. Section-3 describes an overview of the hard timetabling, hdtt. The tabu search with PFP-GA used to solve hdtt problem is presented in section-4. The performance of the algorithms in solving the problem is discussed in section-5. A conclusion and future work are described in section-6 and section-7 respectively.

II. TABU SEARCH AND TIMETABLING

Alberto Colorni et al. [1] presented the model and definition of hierarchical structure for objective function and genetic operators. They compared genetic algorithms with and without local search with manual method and two approaches based on simulated annealing and tabu search. They showed that GAs with local search and tabu search performed better than simulated annealing and manual timetables. Alberto Colorni et al. [2] compared various versions of simulated annealing and tabu search with a genetic algorithm-based approach. They found genetic algorithm produced better timetables than simulated annealing and tabu search had been the best performing algorithm. They did an empirical comparison of metaheuristics on real-world data regarding high-school timetabling, the genetic operators minimizing generalized cost functions, the hierarchical structuring of the objective function, the filtering algorithm. S. C. Chu and H. L. Fang [3] compared tabu search and genetic algorithm approaches for timetabling. They showed that TS approach could find better timetables in lesser time than GA. However, different near optimal solutions could be produced simultaneously using GA.

Mohamed Tounsi and Philippe David [4] presented a framework based on successive use of two local search algorithms to solve real examination timetabling problems. They described how the local search algorithm assisted any other specific local search algorithm to come out from local optimality. Their experiments showed that cooperative algorithm, a tabu search with a local search algorithm, worked well than using tabu search only.

Marcone Jamilson et al. [5] proposed a GRASP (Greedy Randomized Adaptive Search Procedure) algorithm that used a partially greedy procedure to construct an initial solution and attempted to improve it using a tabu search. Haroldo G. Santos et al. [6] presented a new TS heuristic for school timetabling problem. They proposed two different memory based diversification strategies. Experiments with real instances and comparison with previously proposed TS showed that their method produced better solutions with faster times. Edmund K. Burke et al. [7] presented a case-based heuristic selection approach for automated University course and exam timetabling. Knowledge discovery techniques were employed in two distinct scenarios- modeling the problem and the problem solving situations along with specific heuristics for those problems. The case based reasoning could act effectively as an intelligent approach to learn which heuristics worked well for particular situations. They explored approach for case-based heuristic selection to predict the best heuristics for course timetabling problems and the best sequential heuristics for exam timetabling problems. They used tabu search and hill climbing to study the performance within the case based heuristic selection framework.

Marco Chiarandini et al. [8] described and analyzed a hybrid meta-heuristic algorithm combining different construction heuristics, variable neighborhood descent, tabu search, and simulated annealing. They showed that well designed hybrid algorithms led to high performing techniques for hard combinatorial problems. They proposed a framework which successively applied construction heuristics, variable neighborhood descent and simulated annealing. The framework achieved good results on timetabling problems such as employee, examination timetabling.

Khang Nguyen et al. [9] applied a simulated annealing based algorithm with an intensification strategy to a real world high school timetabling problem. They compared proposed algorithm with two known results: handmade timetables and the results obtained from tabu search algorithm. The results were better than the both methods. Nguyen Ba Phuc et al. [10] presented a new hybrid GA-Bees algorithm for solving a real world University timetabling problem in Vietnam. The hybrid approach was tested and compared with VNS (Variable Neighborhood Search), the bees algorithm, and tabu search. The result proved that an approach produced better quality solutions to the problem.
Yong Ou Yang and Yi Chen [11] presented a GA, TS and the coloring principle based course scheduling system. It was mainly based on the hybrid genetic algorithm to achieve the more complex algorithm to solve the actual requirements of the various hard, soft constraints. The testing took long time for automated timetabling, but it was less than manual timetabling and improved efficiency of administrative department.

Meysam Shahvali Kohshori et al. [12] proposed three new hybrid genetic algorithms for solving the University Course Timetabling Problem (UCTP): FGASA (Fuzzy Genetic Algorithm guided by Simulated Annealing), FGARI (Fuzzy Genetic Algorithm guided by Randomized Iterative local search), and FGATS (Fuzzy Genetic Algorithm guided by Tabu Search). Violation of soft constraints in fitness function was measured by fuzzy logic. Simulated annealing, tabu search and randomized iterative local search were applied to improve ability of exploitation and prevent GA to be trapped in local optimum.

III. THE HARD TIMETABLING PROBLEM

We focus on solving the high school timetabling problem hdtt, “hard timetabling” given as test data sets in OR-Library. All timeslots must be used with very less or no options for each assignment. There are currently five data files on timetabling. D. Abramson and H. Dang [13], M. Randall, D. Abramson and C. Wild [14], K. A. Smith, D. Abramson and D. Duke [15] initially used these datasets. The data files describe five timetabling problems: hdtt4, hdtt5, hdtt6, hdtt7, and hdtt8. Each problem consists of three text files.

For hdtt4 these files are: hdtt4list (list of requirements); hdtt4note (dimensions of the problem); hdtt4req (requirements matrix). Table 1 shows the grouping of rows according to venue for hdtt4. If there are C classes, T teachers, V venues, and P periods, then the venue1 block of Table-1 indicates the number of times each class-teacher combination is to meet each other in venue1 across the P periods and so on. It is a five days week, six periods a day with a total of 30 periods. All five problems have the following hard constraints:

•A room must not be allocated more than once to a timetable period.
•A class must not be scheduled more than once in a period.
•A teacher must not be scheduled more than once in a period.
•All class-teacher-venue tuples must be scheduled the required number of times.

Table 1 hdtt4 requirements matrix

class/ teacher	venue1				venue2				venue3				venue4			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	2	2	1	2	2	5	1	2	2	1	1	2	3	1	2	1
2	1	1	1	2	0	4	3	2	0	0	5	1	1	4	1	4
3	1	1	1	6	1	2	1	0	2	1	4	1	3	3	2	1
4	2	2	3	2	2	2	1	2	6	1	2	1	2	0	1	1

IV. TABU SEARCH AND PFP-GA FOR HARD TIMETABLING PROBLEM

We have used a C++ library of genetic algorithm objects, GAlib, developed by Matthew Wall of Massachusetts Institute of Technology [16]. The library includes GA tools to do optimization in any C++ program using any representation and operators, e.g. Simple Genetic Algorithm (SGA) as described in [17] uses non overlapping populations. It creates a new population of individuals per generation, by selecting from the previous population then mating to produce the new offspring. The crossover operator generates a child from two parents. The mutation is applied on each newly generated offspring, based on the mutation probability.

Our objective function returns the fitness score of individual chromosome (timetable) based on error value (error), which is sum of number of workload violations (error1) and number of clashes (error2). It is given by-
Score= [1/ (0.1+error)], hence timetable with zero error value has score 10. Same room or teacher value at the same index in succeeding classes leads to a clash. The values, error1 and error2 are expressed by the following two equations:

$$error1 = \sum |(Wrct - Arct)| \quad \forall\ 0<=r<rooms,\ 0<=c<classes,\ 0<=t<teachers.$$
W=workload, A=allotment (1)

$$error2 = \sum_{l=0}^{classes-2} \sum_{m=l+1}^{classes-1} \sum_{n=0}^{slots-1} 1$$
If ((TT [l*slots+n]) == (TT [m*slots+n])),
Otherwise 0. (2)

Where, TT is individual timetable (chromosome) and slots= (2*days*hours).

4.1 Partial Feasibility Preserving GA

We have modified initialization, crossover and mutation routines to improve the performance of GA. The individuals are initialized with room and teacher values, satisfying workload requirements. Mutation is ***adaptive*** in nature as the probability is decided on number of errors in the individual, more the errors higher is the probability. It gets an individual and checks room/teacher clashes in succeeding classes probabilistically. If it finds a clash then it picks up a number randomly within the range of room/teacher values and replaces the clashing value with this number, without creating another clash.

We have also designed workload preserving crossover i.e. it always maintains workload feasibility during the GA runs. The children are created by copying left part of the respective parent within the respective class to left part of the corresponding class in child. Their right parts are made up of right part of the respective parent. This ***intelligent*** crossover calculates difference between actual requirement and partial allotment while copying second parts to children. If it's nonzero then corresponding room, teacher pairs are copied to right parts of children otherwise the positions are

filled with invalid value -1. According to new difference, pairs are inserted as per actual requirement by replacing -1s. Finally we find the children retaining workload requirements.
The repair function resolves overlaps, if any. It gets a chromosome, the class number and the index n of overlapped value. It tries probabilistically an index k within the class slots so that the swapping of k and n will remove overlap.

4.2 Tabu Search

Initializer provides the initial solution for tabu search algorithm. Specified number of neighbors is created from initial solution by randomly swapping room, teacher pairs within class. If the fitness of new member is less than the swapped neighbor we call the randomizer again to obtain indices for creation of another candidate solution by swapping. The indices of these pairs are stored in list i.e. tabu list, whose size is variable. These locations are avoided while doing swap operation to get next neighbor. We calculate the fitness value of each neighbor using objective function discussed earlier. The best solution amongst neighbors is used as initial solution for next iteration. Repair function within iteration does its job of making the solutions more fit than before. We allow moves from tabu list, if the fitness score of individual i.e. aspiring candidate, with these moves is better than that of best solution in current iteration. Entire process is repeated for given number of iterations. The best solution out of tabu search is retained in an array for future use.

Now, the modified genetic algorithm steps in and continues upto the given number of generations to find an optimal solution. The tabu search outcome, the above said best solution, is used as an input to newly written initializer which creates population of solutions. This helps our GA in exploration and converging much faster than running alone. The method is expressed by Algorithm-1.

```
1. Initialize solution with workload fulfillment
2. Repeat steps 3 to 6 for number of iterations
3. Create specified number of neighbors by randomly swapping room, teacher pairs
   {(if first iteration) from the initial solution, else from the best of previous iteration}
4. Store the indices of the pairs in tabu list of given size; avoid these moves in further swapping
5. Calculate the fitness of individuals; find the best of iteration
6. If moves from tabu list create better individual than the best of current iteration then allow it (Aspiration moves)
7. Use the best solution of last iteration of tabu search to initialize PFP-GA
8. Run PFP-GA for specified number of generations/ till solution is obtained
```

Algorithm-1 TS and PFP-GA

V. RESULTS AND DISCUSSION

Our implementation is in C++ using Dev C++ compiler, version 4.9.9.2 on Intel(R) Core™ 2 Duo CPU 2.4 GHz with 2 GB of memory and Windows Vista. SGA was applied to hdtt4 problem described in section III. Figure 1 shows graphically errors in the timetable after specified generations of SGA. We fixed population size to 100 by performing trial runs. SGA could give solution to hard timetabling, hdtt4 within a few minutes in 80000 generations. Thus SGA could help in finding solution to the problem in finite time.

We have modified the basic GA operators to improve performance i.e. to reduce errors and speed up execution. The Figure 2 shows performance graph of PFP-GA with intelligent operators and repair function. The population size and crossover probability, seed were 100, 1.0 and 100 respectively. An algorithm gave solution in 400 generations and within ***few*** seconds compared to that of SGA (80000 and around 9 minutes) [18].

PFP-GA with tabu search applied on hdtt4 results in 9, 3, 0 errors after 40, 120, 200 generations respectively (Figure 3). We observed significant improvement in execution time and obtained the result in 2500 iterations of tabu search and 200 generations of PFP-GA, within ***seven*** seconds which is worth noting in hard timetabling. It’s less than most of the known times so far, out of different methodologies. Execution time split is shown in Figure 4.

The performance for hdtt4 problem was tested against the comparison given by Nelishia Pillay [19]. The comparison was carried out on machine ***equivalent*** to our configuration. It was about the following methodologies applied to the Abramson benchmark set [15]. GA with SPHH- selection perturbative hyper-heuristic, NN-TT2 and NN-TT3- Neural network approaches, SA1 and SA2- Simulated annealing, TS- Tabu search and GS- Greedy search. The PFP-GA with TS ***outperforms*** all methods (Table 2). It is clear from various runs and subsequent graphical analysis that knowledge augmented operators and repair function help in giving faster solutions for the GA based optimization problems like timetabling.

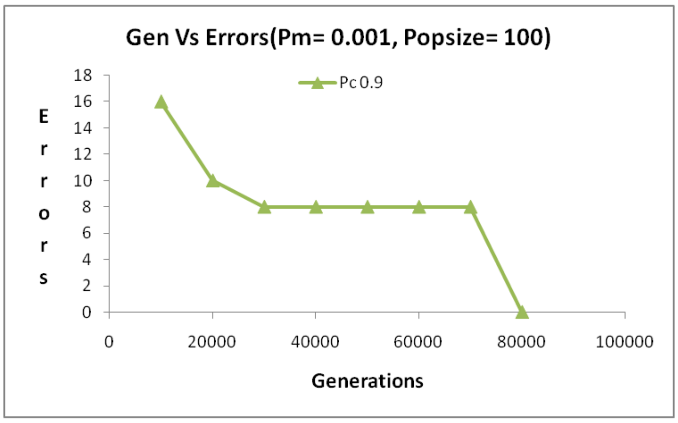

Figure 1 Convergence of SGA (Errors=216 for 0 generation)

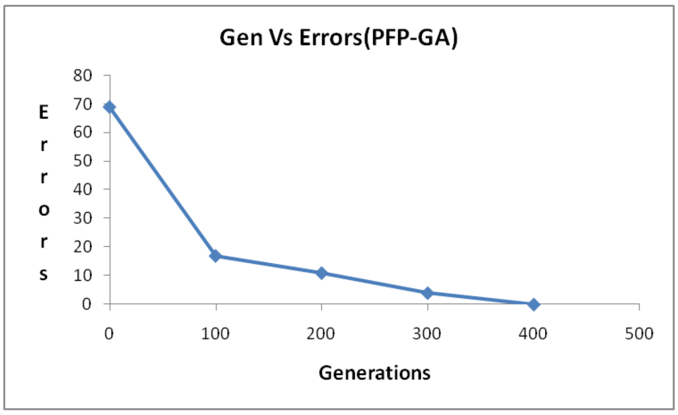

Figure 2 Performance of PPF-GA only (Ps=100, Pc=1.0, seed= 100)

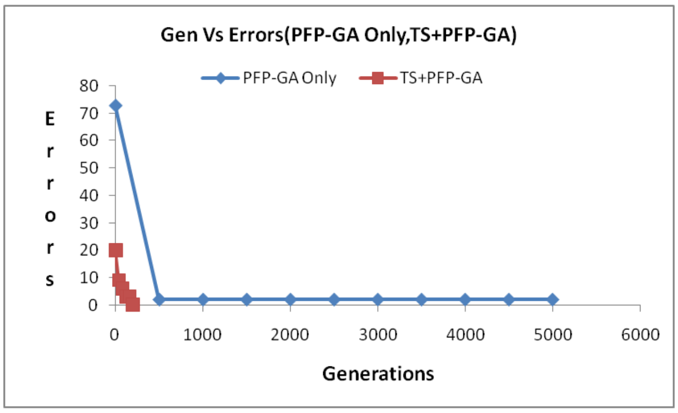

Figure 3 Performance of PFP-GA only (Ps=150, Pc=1.0, seed= 1500) and TS+PFP-GA (20 Errors after 2500 TS iterations)

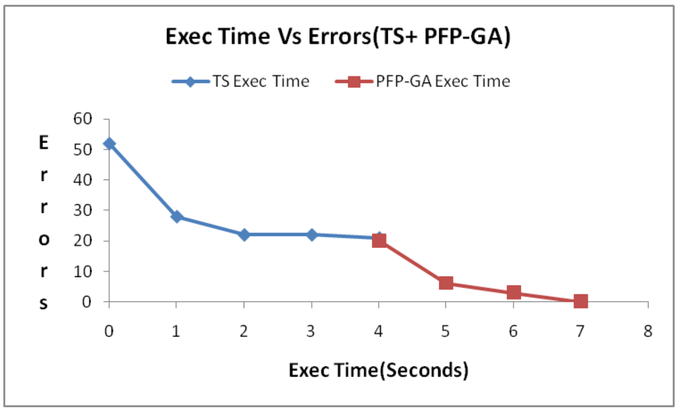

Figure 4 Analysis of execution time (TS+PFP-GA)

Table 2 Performance comparison
(Execution time for hdtt4 in Seconds)

Method	Execution Time
Proposed TS+PFP-GA	**7**
SPHH	12
NN-TT2	29
NN-TT3	109
SA1	-
SA2	15
TS	630
GS	16

VI. Conclusion

We presented a hybrid genetic algorithm to solve high school timetabling problem hdtt, "hard timetabling", specifically hdtt4, given as test data set in OR-Library. It produced a solution that met all the constraints for the problem. We proposed GA whose operators used domain knowledge during the initialization, crossover, and mutation to expedite the search. The probabilistic repair while creating offspring contributed significantly to average fitness of the next population. Furthermore the combination of tabu search and PFP-GA gave results in a few seconds i.e. in much less time compared to SGA. It performed better than most of the known times, given in comparative studies on similar configuration.

VII. Future Work

We will apply the genetic algorithm to additional hdtt and school timetabling problems in future. The hybrid of genetic algorithm and local search techniques like simulated annealing, tabu search etc. will be used to solve complex timetabling instances such as benchmark datasets proposed by international timetabling competitions.

References

[1] Alberto Colorni, Marco Dorigo and Vittorio Maniezzo, "A Genetic Algorithm to Solve the Timetable Problem", computational optimization and applications journal, 1994
[2] Alberto Colorni, Marco Dorigo and Vittorio Maniezzo, "Metaheuristics for Highschool Timetabling", Computational Optimization and Applications -9, pp.275–298, 1998
[3] S. C. Chu and H. L. Fang, "Genetic Algorithms vs. Tabu Search in Timetable Scheduling", Third International Conference on Knowledge-Based Intelligent Information Engineeing Systems, Adelaide, Australia, 1999

[4] Mohamed Tounsi and Philippe David, "Local Search Algorithm to improve the Local Search", Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'02), p 438, 2002
[5] Marcone Jamilson Freitas Souza, Nelson Maculan and Luis Satoru Ochi, "A GRASP-Tabu Search Algorithm for Solving School Timetabling Problems", Metaheuristics: Computer Decision-Making, pp.659-672, Kluwer Academic Publishers, B.V., 2003
[6] Haroldo G. Santos, Luiz S. Ochi, and Marcone J.F. Souza, "An Efficient Tabu Search Heuristic for the School Timetabling Problem", LNCS, 3059, pp.468-481, International Workshop on Experimental and Efficient Algorithms, Springer, 2004
[7] Edmund K. Burke, Sanja Petrovic and Rong Qu, "Case-based heuristic selection for timetabling problems", Journal of Scheduling, Vol. 9, No.2, pp.115-132, 2006
[8] Marco Chiarandini , Mauro Birattari , Krzysztof Socha and Olivia Rossi-Doria, "An effective hybrid algorithm for university course timetabling", Journal of Scheduling, Vol.9, pp.403–432, 2006
[9] Khang Nguyen, Tung Pham, Nga Le, Nguyen Dang and Nuong Tran, " Simulated Annealing-based Algorithm for a Real-world High School Timetabling Problem", IEEE Second International Conference on Knowledge and Systems Engineering, pp.125-130, 2010
[10] Nguyen Ba Phuc, Nguyen Tan Tran Minh Khang and Tran Thi Hue Nuong, "A New Hybrid GA-Bees Algorithm for a Real-world University Timetabling Problem", IEEE International Conference on Intelligent Computation and Bio-Medical Instrumentation, pp.321-326, 2011
[11] Yong Ou Yang and Yi Chen, " Design of automated Course Scheduling system based on hybrid genetic algorithm", The 6th International Conference on Computer Science & Education (ICCSE), pp.256-259, 2011, Singapore
[12] Meysam Shahvali Kohshori, Dariush Zeynolabedini, Mehrnaz Shirani Liri and Leila Jadidi, "Multi Population Hybrid Genetic Algorithms for University Course Timetabling Problem", International Journal of Information Technology and Computer Science, Vol.4, No.6, pp.1-11, 2012
[13] D. Abramson and H. Dang, "School timetables: a case study in simulated annealing: sequential and parallel algorithms", Lecture Notes in Economics and Mathematics Systems, V.Vidal (Ed.), Springer-Verlag: Berlin, pp. 103-124, 1993
[14] M. Randall, D. Abramson and C. Wild, "A general meta-heuristic based solver for combinatorial optimization problems", Technical report TR99-01, School of Information Technology, Bond University, Queensland, Australia
[15] K. A. Smith, D. Abramson and D. Duke, "Hopfield neural networks for timetabling: formulations, methods, and comparative results", Computers and Industrial Engineering, Vol. 44, No. 2, pp. 283-305, 2003
[16] Matthew Wall, "GAlib: A C++ Library of Genetic Algorithm Components" ,version 2.4,Documentation Revision B, August, 1996,(http://lancet.mit.edu/ga/),Massachusetts Institute of Technology (MIT) (c) 1995-1996, Matthew Wall (c) 1996-2005
[17] D.E.Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning", Addison-Wesley, 1989
[18] Sanjay R. Sutar and Rajan S. Bichkar, "Genetic Algorithms based Timetabling using Knowledge Augmented Operators", International Journal of Computer Science and Information Security, (1947-5500), pp.570-579, Vol.14, No.11, 2016
[19] Nelishia Pillay, "A Comparative Study of Hyper-Heuristics for Solving the School Timetabling Problem", Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference, (SAICSIT'13), pp. 278-285, 2013

AUTHORS BIOGRAPHY

Sanjay R. Sutar was born in Satara, India, in 1971. He received the Bachelor degree from the Shivaji University Kolhapur, India in 1992 and the Master degree from the Dr. B. A. Tech. University, Lonere, India in 2003, both in Computer Engineering. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, S.R.T.M. University, Nanded, India. His research interests include automated scheduling and evolutionary algorithms.

Rajan S. Bichkar was born in Satara, India, in 1966. He received the Bachelor degree in 1986 and the Master degree in 1991 both in Electronics Engineering from the B. A. Marathwada University Aurangabad, India and the Ph.D. degree in 2000 from Indian Institute of Technology, Kharagpur, India. His research interests include image processing and evolutionary algorithms.