Solving Class Timetabling Problem of IIT Kanpur using Multi-Objective Evolutionary Algorithm

Dilip Datta, Kalyanmoy Deb
KanGAL, Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur - 208 016, India
{ddatta,deb}@iitk.ac.in

Carlos M. Fonseca
DEEI, Faculty of Science & Technology,
University of Algarve, Campus de Gambelas
8000-117 Faro, Portugal
cmfonsec@ualg.pt

KanGAL Report Number 2006006

Abstract

Unlike in many other universities, preparation of class timetable in IIT Kanpur is very laborious and complicated. It contains different types of classes, among which most of the common classes are either split or grouped. Many split classes are divided up to five parts, while many sets of group classes contain up to twenty classes. The entire timetable is composed of two phases. The first phase contains all the common compulsory classes of the institute, which are scheduled by a central team. The second phase contains the individual departmental classes. Presently this timetable is prepared manually, by manipulating those of earlier years, with the only aim of producing a feasible timetable. The potentiality of evolutionary algorithms (EAs) have been exploited in the present work to schedule the classes of the first phase of the problem. Using NSGA-II-UCTO, a multi-objective EA-based university class timetable optimizer, a number of trade-off solutions, in terms of multiple objectives of the problem, could be obtained very easily. Moreover, each of the obtained solutions has been found much better than a manually prepared solution which is in use.

Keywords: university class timetabling, multi-objective optimization, evolutionary algorithms, NSGA-II, NSGA-II-UCTO.

1 Introduction

The class[1] timetabling problem is a typical scheduling problem that appears to be a tedious job in every academic institute once or twice a year. The problem involves the

[1] A class is a meeting of students and teacher in a room for a lecture.

scheduling of classes, students, teachers and rooms at a fixed number of time-slots, subject to a certain number of constraints. An effective timetable is crucial for the satisfaction of educational requirements, and the efficient utilization of human and space resources, which make it an optimization problem. Traditionally, the problem is solved manually by *trial and hit* method, where a valid solution is not guaranteed. Even if a valid solution is found, it is likely to miss far better solutions. These uncertainties have motivated for the scientific study of the problem, and to develop an automated solution technique for it. The problem is being studied for last more than four decades, but a general solution technique for it is yet to be formulated.

The problem was first studied by Gotlieb [16], who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students, one teacher, and any number of times which could be chosen freely. Since then the problem is being continuously studied using different methods under different conditions. Initially it was mostly applied to schools (de Gans [14], de Werra [15], Lawrie:1969 [17], Schaerf [23], Tripathy [25]). Since the problem in schools is relatively simple because of their simple class structures, classical methods, such as linear or integer programming approaches (Lawrie [17], Tripathy [25]), could be used easily. However, the gradual consideration of the cases of higher secondary schools and universities, which contain different types of complicated class-structures, is increasing the complexity of the problem. As a result, classical methods have been found inadequate to handle the problem, particularly the huge number of integer and/or real variables, discrete search space and multiple objective functions. These inadequacy of classical methods have drawn the attention of the researchers towards the heuristic-based non-classical techniques. Worth mentioning non-classical techniques, that are being used to the problem, are genetic algorithms (Colorni et al. [7; 8], Abramson and Abela [1]), neural network (Looi [19]), and tabu search algorithm (Costa [9]). However, compared to other non-classical methods, the widely used are the genetic/evolutionary algorithms (GAs/EAs). The reason might be their successful implementation in a wider range of applications. Once the objectives and constraints are defined, EAs appear to offer the ultimate *free lunch* scenario of good solutions by evolving without a problem solving strategy (Al-Attar [2]). A few worth mentioning EAs, used to school timetabling problem, are those of Abramson and Abela [1], Piola [21], and Bufé et al. [4]. Similarly, EAs, used to university class timetabling problem, are those of Lima et al. [18], Carrasco and Pato [5], Srinivasan et al. [24], Blum et al. [3], Rossi-Doria and Paechter [22], and Datta et al. [10].

Datta et al. [10] modelled the university class timetabling problem as a multi-objective optimization problem, considering different class-structures, such as single-slot, multi-slot, split, combined, open, and group classes. NSGA-II-UCTO, a version of EA-based multi-objective optimizer NSGA-II (Deb [12], Deb et al. [13]), was also developed to handle the problem, and demonstrated successfully in two real problems. Since their model resembles with the class timetabling problem of Indian Institute of Technology Kanpur (IIT Kanpur), NSGA-II-UCTO has been applied to schedule the common compulsory even-semester classes of IIT Kanpur. Using NSGA-II-UCTO, a number of trade-off solutions, in terms of multiple objectives of the problem, could be obtained very easily. Moreover, each of the obtained solutions has been found much better than a manually prepared solution which is in use.

2 Even-Semester Class Timetabling Problem of IIT Kanpur

There are two academic sessions in IIT Kanpur in a year: odd-semester and even-semester. Odd-semester is run in the months of July-November, and covers the classes of odd numbered semesters, like first, third, fifth and seventh semesters. While even-semester is run in the months of December-May, and covers the classes of even numbered semesters, like second, fourth, sixth and eighth semesters. Class timetable of each of odd and even-semesters is prepared manually, by manipulating those of earlier years, with the only aim of producing a feasible timetable. Each class timetable is composed of two phases. The first phase contains the common compulsory classes of all under-graduate programmes (B.Tech and integrated M.Sc), and the timetable of this phase is prepared by a central team. Then the available time-slots and rooms are allotted to different departments to prepare the second phase of the timetable. In this phase, departments need to schedule independently their departmental classes, which are offerred to their B.Tech and integrated M.Sc programmes, and also to other Masters and Ph.D programmes. The scheduling of even-semester classes has been considered in the present work, and it has been named as IITK2 in short. In addition to the first phase of common compulsory classes of even-semesters of all B.Tech and integrated M.Sc programmes, slots for departmental compulsory and elective classes of Mechanical Engineering department, for its B.Tech programme, have also been included in IITK2. This inclusion is only for illustrative purpose to show that all common compulsory classes, as well as other departmental compulsory and open classes, can also be scheduled by a single team. Class-structures of IITK2 are very complex, which include different types of classes, such as single-slot, multi-slot, split, combined, open, and group classes. Laboratory classes are spanned over two or three consecutive time-slots, and many of them are split up to 5 parts. Most of the tutorial classes are grouped, and few groups contain up to 20 classes. Total number of only common compulsory courses[2] is 125. These courses involve 242 classes which span over 266 time-slots. Detail of these classes is shown in Table 1. Classes of IITK2 are to be taught to around 2000 by 103 teachers, and these

Table 1: Common compulsory classes of IITK2.

Classes	**Number of Classes**				**No. of Slots**
	Simple	**Split**	**Combined**	**Open/Group**	
1-Slot	11	–	–	219	230
3-Slot	–	12	–	–	36
Total	**11**	**12**	**–**	**219**	**266**

are required to be scheduled in 40 rooms (including laboratories) in 5 days/week, where each day has 8 time-slots with a recess after the 5-th time-slot. Moreover, the timetable is subject to the following six types of *hard constraints*, which must be satisfied by a solution to accept it as a valid one:

1. A student should have only one class at a time.
2. A teacher should have only one class at a time.
3. A room should be booked only for one class at a time (a set of combined classes may be treated as a single class).

[2]A course is a subject to be studied, for example, *Theory of Optimization*, or *Introduction to Numerical Methods*.

4. Only one class of a course should be scheduled on a day.

5. A class should be scheduled only in a specific room, if required, otherwise in a general room which has sufficient sitting capacity for the students of the class. Due to the requirement of some extra facilities, such as laboratory apparatus, many classes may need to be scheduled only in specific rooms.

6. A class should be scheduled only at a specific time-slot, if required. Due to many reasons, such as involvement of senior teachers in administrative works, some classes may need to be scheduled only at specific time-slots.

As per the formulation of Datta et al. [10], the total number of hard constraints under such conditions, involved only with common compulsory classes, is (S+M+R)TD+CD+3E = 47071, where S, M, R, T, D, C and E represent, respectively, the total numbers of students, teachers, rooms, time-slots/day, days/week, courses, and classes. However, since only common compulsory classes of all B.Tech and integrated MSc programmes have been considered here, which are taught in the second and fourth semesters only, computational time can be reduced by replacing the student-clash constraints by batch-clash constraints. This is possible due to the reason that, in the absence of any open classes, a batch is free/engaged at a time-slot means all the students of the batch are also free/engaged at that time-slot. In that case, the total number of hard constraints in IITK2, involved with common compulsory classes, is reduced from 47071 to 7151.
In addition to hard constraints, class timetabling problem is generally considered under some *soft constraints*. These constraints do not represent any physical conflict, but preferences only that are encouraged to be fulfilled whenever possible (Melício et al. [20], Bufé et al. [4], Carrasco and Pato [6]). Following three types of soft constraints have been considered in IITK2:

1. Students should not have any free time-slot between two classes on a day.

2. Classes of teachers should be well spread over the week.

3. A smaller class should not be scheduled in a room which can be used for a bigger class.

The soft constraint (1) implies a compact timetable, whereas the constraint (2) conflicts with it, and seeks a well-spread timetable. The constraint (3) takes care of proper utilization of rooms. Since soft constraints are preferences only, more such constraints can be imposed without any loss of generality of the problem. However, the imposition of a constraint will force a solution to respect it, and hence, the solution may get altered by the imposition of each soft constraint.
Though class timetabling problem is tackled as an optimization problem, it does not have any fixed objective function to optimize. An objective function in this problem is just an arbitrary measure of the quality of a solution (Abramson and Abela [1]). Hence, the choice of objectives varies from university to university. Most of the researchers treated the problem as a single-objective optimization problem, and took the minimization of total constraint violation as the only objective function (Abramson and Abela [1]; Blum et al. [3]; Lima et al. [18]; Piola [21]). However, various choices of objectives, such as a compact or spread timetable, lead the problem to a multi-objective optimization problem. The two conflicting soft constraints, constraints (1) and (2), have been considered in IITK2 as two objective functions for optimizing during the optimization process, i.e,. the objective functions are:

1. Minimize the average number of weekly free time-slots between two classes of a student (f_1).

2. Maximize the average of weekly span of time-slots of classes of a teacher (f_2).

In f_1, classes only of the same day, not of different days, are considered in finding the free time-slots of a student. Since objective functions in the problem are nothing but functions of the soft constraints imposed on it, without any loss of generality of the problem, different objectives can be incorporated by imposing different soft constraints on the problem.

3 NSGA-II-UCTO: NSGA-II as University Class Timetable Optimizer

NSGA-II-UCTO, developed by Datta et al. [10] as a multi-objective EA-based university class timetable optimizer, is stated briefly in this section. The basic component of an EA is chromosome which represents a solution in the search space of an optimization problem. A chromosome is composed of genes, each of which describes a parameter of a problem. A set of chromosomes forms a population for an EA, evolution of which takes place over the repeated application of EA operators, particularly selection, crossover and mutation. The function of selection operator is to emphasize good solutions and eliminate weak solutions. Crossover and mutation operators are responsible for the generation of offspring (new solutions). The salient features of NSGA-II-UCTO, used to solve IITK2, are as given below:

1. **Chromosome Representation** is a two-dimensional matrix, each column of which represents a time-slot, and a row represents a room. Then, the value of each cell of the matrix represents the class(s), scheduled in the corresponding room and time-slot. N such chromosomes are used to form the population of NSGA-II-UCTO.

2. **Heuristic Approach** is used for initializing the chromosomes of the population. In this approach, classes are first sorted in descending order of their complexities, and then time-slots and rooms are allotted to them.

3. **Crowded Tournament Selection Operator** (Deb [12]) is used to form a *mating pool* (Deb [11]) of N solutions from the population. It is done by randomly selecting two solutions from the population, and sending a copy of the best one, based on ranks and crowding distances (Deb [12]), to the mating pool. The process is continued until the mating pool is filled up with N solutions. The mating pool is later used by EA operators for generating offspring.

4. **Crossover for Valid Resource Allocation (XVRA)** is used for generating a new population of N offspring. In crossover, two random solutions are picked up from the mating pools, and two offspring are generated by exchanging some randomly selected information from the chosen solutions.

5. **Mutation for Reshuffling Resource Allocation (MRRA)** is used for mutating the offspring of the new population, where information of two random slots of a chromosome are interchanged.

6. Both the populations, obtained so far, are combined to form a combined population of 2N solutions.

7. Based on ranks and crowding distances, the best N solutions from the combined population are picked up to form a single population.

8. Steps (3)-(7) are repeated for required number of generations (iterations).

9. Result obtained after the required number of generations is accepted as the optimum result.

4 Solution of IITK2 using NSGA-II-UCTO

It was shown by Datta et al. [10] that NSGA-II-UCTO is unable to handle relaxed constraints. Hence, no such attempt has been made in case of IITK2, but it has been solved directly maintaining feasibility of solutions. Presently the timetable of IITK2 is prepared manually by *trial and hit* method with the only aim of producing a feasible timetable. Applying NSGA-II-UCTO, not only feasible and trade-off solutions, but much better results than the manually prepared one, could be obtained. A comparison plot of the results for common compulsory classes of IITK2 is shown in Figure 1. Points A

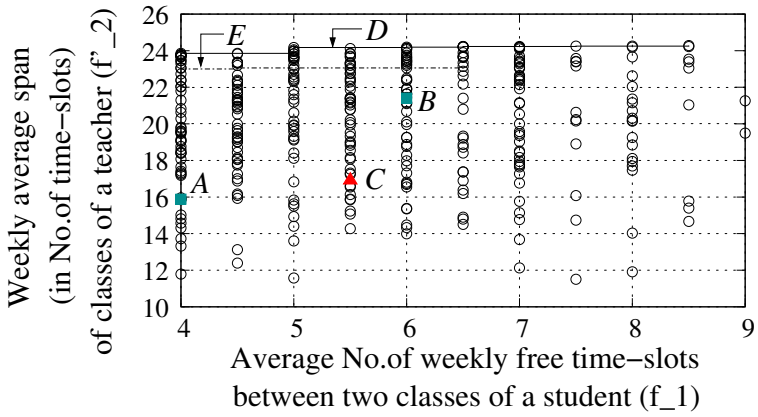


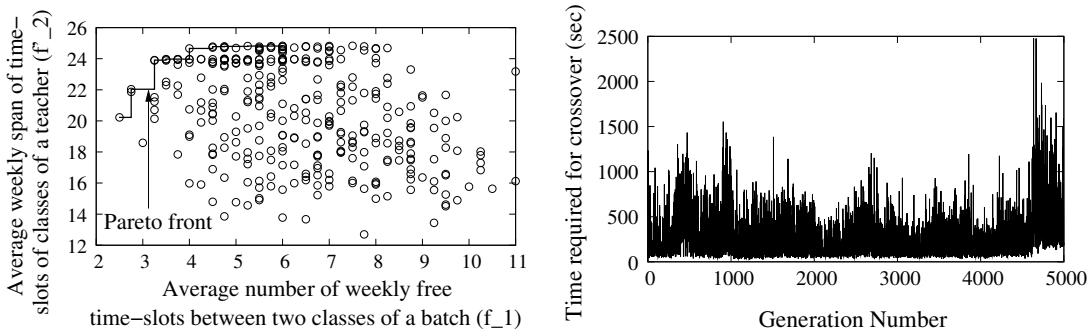
Figure 1: Comparison of scheduling of common compulsory classes of IITK2. A and B: Single-objective optimization of f_1 and f_2, respectively; C: Manually prepared timetable which is in use; D: Multi-objective optimization with $p_c = 0.90$, evolving p_m and $r_s = 0.125$, and E: Multi-objective optimization under MRRA alone with evolving p_m and $r_s = 0.125$.

and B represent, respectively, single-objective optimization of f_1 and f_2. Curve D is the Pareto front of multi-objective optimization with crossover probability $(p_c) = 0.90$, evolving mutation probability (p_m) and random seed $(r_s) = 0.125$, while curve E is that with MRRA alone under evolving p_m and $r_s = 0.125$. Point C is the manually prepared result which is in practice. It is observed that both the single-objective (A and B) and multi-objective (D and E) optimization results are better than the manually prepared result (C). In case of multi-objective optimization, variation in objective function f_2 is very small. This might be due to the fact that most of the classes of IITK2 are grouped, for which it is very tough to shift a particular group of classes in different slots. One solution, produced by NSGA-II-UCTO, is also shown in Table 2 (rooms are not shown in the Table).

Plots of one run for complete IITK2 are shown in Figure 2. It is observed from Figure 2(b) that on an average of 5 minutes 35 seconds per generation has been required by crossover operator, and total execution time of 465 hours 14 minutes 39 seconds has been required,

Table 2: Common compulsory classes of IITK2.

	08.00	**09.00**	**10.00**	**11.00**	**12.00**	**14.00**	**15.00**	**16.00**
M		MTH102NA			MTH012NB			MTH102NAT
	MTH102NBT	PHY103NB	D0	ESC101NT	PHY103NA			PHY103NBT
O	PHY103NAT	MTH101S	ESO-I(T)	ESC102NT	PHY102S	CHM101LabT	ESC101N	MTH101ST
	PHY102ST	MTH101R	CHM201RT	HSS-I	PHY102R	D2/OE	ESC102N	MTH101RT
N	PHY103RT	ESO-II			ESO-I			ESO-II
		MTH203R			CHM201R			MTH203RT
T	CHM101Lab2							
	PHY101Lab1			ESC101N		ESC101NLab5		
U	ESC101NLab3			ESC102N	D2/OE	ESO214Lab1		
	PHY102Lab4			HSS-1				
E	ESO210Lab1							
W					MTH102NA	D3/OE	ESO-II	HSS-I
	ESC102NLab5			D0	PHY103NB		MTH203R	
E	ESO210Lab2			ESO-I	MTH101S	PHY101Lab3		
				CHM201R	MTH101R	ESC101NLab2		
D					D2/OE	ESC102NLab1		
Th	CHM101Lab3				MTH102NB			
	ESC101NLab1			D0	PHY103NA			
H	ESC102NLab2			HSS-1	PHY102S	ESO210Lab3		
	ESO214Lab2				PHY102R			
U					D3/OE			
F	CHM101Lab1			MTH102NBT		MTH102NA	MTH102NB	MTH102NAT
	PHY101Lab2			PHY103NAT	ESC101N	PHY103NB	PHY103NA	PHY103NBT
R	ESC101NLab4			PHY102ST	ESC102N	MTH101S	PHY102S	MTH101ST
	ESC102NLab3			PHY102RT	D3/OE	MTH101R	PHY102R	MTH101RT
I	ESO214Lab2			D2/OE		ESO-I	ESO-I(T)	ESO-II
						CHM201R	CHM201RT	MTH203R



(a) Final solutions

(b) Time for mutation

Figure 2: Solutions of IITK2 ($p_c = 0.90$, $p_m = 0.010$ and $r_s = 0.125$).

for 5000 generations, in Linux environment in a Pentium IV machine with 1.0 GB RAM and 2.933 GHz processor. When solved the problem using mutation operator alone (without crossover operator), total execution time for 5000 generations has come down only to 11 hours 31 minutes 42 seconds. However, the performance of NSGA-II-UCTO has been found better with combined XVRA and MRRA than with MRRA alone (plots are not shown here).

5 Conclusions

Due to the existence of different types of complex classes, preparation of class timetable for IIT Kanpur becomes very laborious and complicated. Presently it is prepared manually, by trial and hit method, with the only aim of producing a feasible solution. The potentiality of evolutionary algorithms (EAs) have been exploited in the present work to schedule the even-semester classes of the institute. Using NSGA-II-UCTO, a multi-objective EA-based university class timetable optimizer, a number of trade-off solutions, in terms of multiple objectives of the problem, could be obtained very easily. Moreover, much better results, than the manually prepared one, have been obtained using NSGA-II-UCTO.

Acknowledgement: This work was partially supported by an Indo-Portuguese scientific and technical cooperation grant from DST, India, and GRICES, Portugal.

References

[1] Abramson, D., & Abela, J., A parallel genetic algorithm for solving the school timetabling problem. In Proceedings of 15 Australian Computer Science Conference, Hobart, 1992, 1-11.

[2] Al-Attar, A., White Paper: A hybrid GA-heuristic search strategy. AI Expert, USA, 1994.

[3] Blum, C., Correia, S., Dorigo, M., Paechter, B., Rossi-Doria, O., & Snoek, M., A GA evolving instructions for a timetable builder. In Proceedings of the Practice and Theory of Automated Timetabling (PATAT'02), 2002, 120-123.

[4] Bufé, M., Fischer, T., Gubbels, H., Häcker, C., Hasprich, O., Scheibel, C., Weicker, K., Weicker, N., Wenig, M., & Wolfangel, C., Automated solution of a highly constrained school timetabling problem - preliminary results. EvoWorkshops-2001, Como, Italy, 2001, 431-440.

[5] Carrasco, M.P., & Pato, M.V., A multiobjective genetic algorithm for the class/teacher timetabling problem. In Proceedings of the Practice and Theory of Automated Timetabling (PATAT'00), Lecture Notes In Computer Science, Springer, 2001, 2079, 3-17.

[6] Carrasco, M.P., & Pato, M.V., A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem. European Journal of Operational Research, 2004, 153(1), 65-79.

[7] Colorni, A., Dorigo, M., & Maniezzo, V., Genetic algorithms and highly constrained problems: The time-table case. In Proceedings of the first International Workshop on Parallel Problem Solving from Nature (PPSN-1, 1990), Lecture Notes in Computer Science (1991), Springer-Verlag, 1990, 496, 55-59.

[8] Colorni, A., Dorigo, M., & Maniezzo, V., A genetic algorithm to solve the timetable problem. Tech. rep. 90-060 revised, Politecnico di Milano, Italy, 1992.

[9] Costa, D., A tabu search algorithm for computing an operational timetable. European Journal of Operational Research, 1994, 76(1), 98-110.

[10] Datta, D., Deb, K., & Fonseca, C.M., Multi-objective evolutionary algorithm for university class timetabling problem, In Evolutionary Scheduling, Springer-Verlag (in Press), 2006.

[11] Deb, K., Optimization for Engineering Design-Algorithms and Examples. Prentice-Hall of India Pvt. Ltd., New Delhi, India, 1995.

[12] Deb, K., Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons Ltd, Chichester, England, 2001.

[13] Deb, K., Agarwal, S., Pratap, A., & Meyarivan, T., A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 2002, 6(2), 182-197.

[14] de Gans, O.B., A computer timetabling system for secondary schools in the Netherlands, European Journal of operations Research, 1981, 7, 175-182.

[15] de Werra, D., Construction of school timetables by flow methods, INFOR - Canadian Journal of Operations Research and Information Processing, 1971, 9, 12-22.

[16] Gotlieb, C.C., The construction of class-teacher timetables. In Proceedings of IFIP Congress, North-Holland Pub. Co., Amsterdam, 1962, 73-77.

[17] Lawrie, N.L., An integer programming model of a school timetabling problem. The Computer Journal, 1969, 12, 307-316.

[18] Lima, M.D., de Noronha, M.F., Pacheco, M.A.C., & Vellasco, M.M.R., Class scheduling through genetic algorithms. IV Workshop do Sistema Brasileiro de Technologia de Informação (SIBRATI), Poli/USP-Säo Paulo, 2001.

[19] Looi, C., Neural network methods in combinatorial optimization. Computers and Operations Research, 1992, 19(3/4), 191-208.

[20] Melício, F., Caldeira, J.P., & Rosa, A., Two neighbourhood approaches to the timetabling problem. In Proceedings of the Practice and Theory of Automated Timetabling (PATAT'04), 2004, 267-282.

[21] Piola, R., Evolutionary solutions to a highly constrained combinatorial problem. In Proceedings of IEEE Conference on Evolutionary Computation (First World Congress on Computational Intelligence), Orlando, Florida, 1994, 1, 446-450.

[22] Rossi-Doria, O., & Paechter, B., An hyperheuristic approach to course timetabling problem using an evolutionary algorithm. The first Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA), 2003.

[23] Schaerf, A., Tabu search techniques for large high-school timetabling problems. In Proceedings of thirteenth National Conference of the American Association for Artificial Intelligence (AAAI-1996), AAAI Press/MIT Press, 1996, 363-368.

[24] Srinivasan, D., Seow, T.H., & Xu, J.X., Automated time table generation using multiple context reasoning for university modules. In Proceedings of IEEE International Conference on Evolutionary Computation (CEC'02), 2002, 1751-1756.

[25] Tripathy, A., School timetabling - A case in large binary integer linear programming. Management Science, 1984, 30(12), 1473-1489.