Innovative Systems Design and Engineering
ISSN 2222-1727 (Paper) ISSN 2222-2871 (Online)
Vol.6, No.2, 2015

A hybrid algorithm for university course timetabling problem

Rakesh P. Badoni* and D.K. Gupta

Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur–721302, India

* E-mail of the corresponding author: rakeshbadoni@gmail.com

Abstract

A hybrid algorithm combining the genetic algorithm with the iterated local search algorithm is developed for solving university course timetabling problem. This hybrid algorithm combines the merits of genetic algorithm and iterated local search algorithm for its convergence to global optima at the same time avoiding being get trapped into local optima. This leads to intensification of the involved search space for solutions. It is applied on a number of benchmark university course timetabling problem instances of various complexities.

Keywords: timetabling, optimization, metaheuristics, genetic algorithm, iterative local search

1. Introduction

Timetabling is an important practical problem that is frequently encountered in educations, enterprises, sports, transportation etc. These combinatorial optimization problems are high dimensional, multi-objectives and belonging to a class of NP-complete problems. A general and effective solution for timetabling is very difficult as it involves problem diversity, constraints variances and changed requirements. Constraints involved can be classified as hard and soft constraints. Hard constraints are those which cannot be violated under any circumstances whereas soft constraints can be relaxed. Soft constraints violations can be permissible, if necessary, but each is penalized with some penalty cost. The university course timetabling problems (UCTPs) schedule a set of events (lectures, tutorials, laboratories, etc.) into a limited number of timeslots and suitable rooms such that the possibility of allocations is maximized and the violation of soft constraints is minimized. Thus, the aim of timetabling problem is usually to obtain a feasible solution by satisfying all the hard constraints and minimize the overall penalty cost of soft constraints violations. In recent years, UCTPs are extensively studied by a number of researchers (Wijaya & Manurung 2009, Yang & Jat 2011, Ayob & Jaradat 2009, Badoni et al. 2014) and a number of approaches are proposed for their solutions. They have systematically categorized these problems, presented their mathematical formulations and described both exact and heuristic approaches for their solutions. Some of the most important methods used are sequential methods, cluster methods and generalized search methods. In (Wijaya & Manurung 2009, Abdullah & Turabieh 2008), these problems are converted into graphs in which the nodes and edges correspond to lectures and constraints and then graph coloring algorithms are used for their solutions. These graph coloring algorithms show a great efficiency in small instances of timetabling problems, but are not efficient in large instances. In constraint based techniques (Wijaya & Manurung 2009), these problems are represented as the Constraint Satisfaction Problems (CSPs) and then solved by using CSPs solving techniques. Several metaheuristic approaches inspired from nature and apply nature-like processes to solutions or populations of solutions are used to get optimal solutions of these problems are Genetic algorithm (GA) (Yang & Jat 2011, Badoni et al. 2014), Tabu Search (TS) (Rossi-Doria et al. 2003, Turabieh & Abdullah 2009), Iterated local search (ILS) (Rossi-Doria et al. 2003), Simulated Annealing (SA) (Rossi-Doria et al. 2003) and Ant Colony Optimization (ACO) (Ayob & Jaradat 2009). In general, there are two types of metaheuristics algorithms known as local area based algorithms and population based algorithms. Local area based algorithms emphasize on exploitation rather than exploration. This means that they move in one direction without performing a wider scan of the search space. Population based algorithms are good at exploration rather than exploitation. In Rossi-Doria et al. (2003), the performances of five different metaheuristics used to solve a UCTP are compared unbiasedly. A guided search genetic algorithm for the solutions of UCTP is discussed in Yang & Jat (2011). The main drawback of these types of algorithms is that they require more time. Thus, hybrid algorithms using two phases, the construction phase finding the feasibility and the improvement phase optimizing soft constraints without coming out of feasible regions of the search space may be a more promising approach.

The purpose of this paper is to develop a hybrid algorithm GAILS combining the GA with the ILS algorithm for the solution of UCTPs. The disadvantage of GA is that it fails to converge to an optimal solution due to repeated searching of different subparts of search space leading to an exponential amount of execution time. Moreover, it uses a local search (LS) algorithm that may fall quickly into a local optimal solution. Thus, when the search space is large, GA may fail to converge or take long execution time as it may get trapped into local optima.

GAILS combines the merits of GA and ILS for its convergence to global optima at the same time avoiding being get trapped into local optima. This leads to intensification of the involved search space for solutions. It is applied on a number of benchmark UCTPs of various complexities. The fitness function is used as a performance measure of the algorithm. Each problem is run for a number of times and the least values of the fitness functions are computed. The fitness functions obtained by GAILS and some other existing algorithms are compared. It is observed that GAILS gives promising results in all considered problem instances.

This paper is organized as follows. Section 1 is the introduction. In Section 2, the UCTP is described. In Section 3, the proposed GAILS algorithm combining GA and ILS to get an optimal solution for the UCTP is described. In Section 4, the implementation and testing of GAILS for a number of problem instances of UCTPs of various complexities are carried out. Finally, conclusions are included in Section 5.

2. University course timetabling problems

In this section, UCTP is described. It is a multidimensional assignment problem, in which events (lectures, tutorials, laboratories, etc.) taken by students are to be scheduled into a number of limited timeslots and rooms in such a way that the violation of predefined set of constraints is minimum. It is assumed that each student takes a number of events and each room is of specified size with a set of features. The problem consists of a set E of n events assigned into 45 timeslots (9 timeslots per day for 5 days), a set R of m fixed seating capacity rooms where events can occur, a set S of p students selecting any event from E and a set F of q room features required by events in selected rooms. The hard constraints considered for this problem are as follows.

1) Only one event is attended by a student at any timeslot.
2) All events are to be assigned to rooms having adequate seating capacity and all the required features.
3) Only one event is assigned to any one room in any timeslot.

The considered soft constraints are as follows.

1) Events should not be scheduled in the last timeslot of a day.
2) No student should attend more than two events in consecutive time slots in a day.
3) All students should have more than one event in a day.

The objective is to satisfy all hard constraints and minimize the violations of the soft constraints in order to get an optimal solution for UCTP. A direct solution representation is chosen.

3. A hybrid algorithm

In this section, a hybrid algorithm GAILS combining GA with ILS is developed for finding optimal solutions of the UCTP described in Section 2. GAILS tries to reduce the exponential time complexity of GA by combing it with an improved version of LS algorithm known as ILS and maximizes the chance of convergence to an optimal solution through using various search spaces. Thus, ILS refines the GA search through successive iterations and minimizes the chance of being getting trapped into local optimal solution. This allows us to take advantage of ILS features in order to improve the population generated by the GA and thus to complement the genetic search.

GA (John, 1992) has been successfully used to solve a large number of combinatorial optimization problems. In this paper, we have used a basic implementation of GA that uses only the problem specific heuristic information. It is characterized by a steady-state evolution process, i.e. at each generation only one couple of parent individuals is selected for reproduction. Tournament selection strategy is used. The initial population is built randomly by assigning a timeslot to each event for each individual using uniform distribution. A uniform crossover operator is used on the solution representation, where for each event; a timeslot's assignment is inherited either from the first or the second parents with equal probability. Mutation is just a random move in the neighborhood defined by the local search extended with three-cycle permutations of the timeslots of three distinct events, which corresponds to the complete neighborhood of the local search. This complete neighborhood is defined as the union of three types of neighborhood moves. Type 1 move takes one event from a timeslot to a different timeslot, type 2 move swaps two events in two different timeslots and type 3 move permutes three events in three distinct timeslots in one of the two possible ways. The offspring replaces the worst member of the population at each generation. The LS algorithm is then applied to each individual. The LS algorithm considered in this paper is same as given in Badoni et al. (2014). The termination criterion of the algorithm is either time limit, or number of iterations, or optimal solution achieved with zero fitness value. In

Algorithm 1, we incorporate ILS into this GA general scheme. The fitness function $f(\mathrm{I})$ for an individual solution I is given by, $f(\mathrm{I}) = \gamma \times \mathrm{hcv}(\mathrm{I}) + \mathrm{scv}(\mathrm{I})$, where hcv(I), scv(I) and γ are the number of hard constraint violations, the number of soft constraint violations and a constant that is always set larger than the maximum possible number of soft constraint violations respectively. In our experiment, the value of constant γ is taken as 10^6. Hence, if $f(\mathrm{I}) \geq 10^6$, solution I will be considered as infeasible.

Algorithm 1 Proposed hybrid algorithm (GAILS)

Input: A problem instance I;

Output: an optimal solution y_{best} for I.

1: **begin**
2: **for** ($i \leftarrow 1$ to *max*) **do //** generate an initial random population of solutions of size *max*.
3: $y_i \leftarrow$ random initial solution;
4: $y_i \leftarrow$ solution after applying LS;
5: calculate fitness function value of y_i;
6: sort population of solutions based on increasing order of their fitness function values;
7: $y_{best} \leftarrow y_1$; // y_1 is the best solution in the population.
8: **repeat**
9: select two parents from population by tournament selection;
10: $y \leftarrow$ child solution created after crossover with probability α;
11: $y \leftarrow$ child solution created after mutation with probability β;
12: **if** ($f(y) < f(y_{best})$) **then //** ($f(y)$ is the fitness function value of y.
13: $y \leftarrow$ solution created after applying the ILS; // ILS given in Algorithm 2.
14: $y_{max} \leftarrow y$; // y replaces the worst solution y_{max} in the population of sorted solutions.
15: generate population of solutions sorted based on increasing order of their fitness value;
16: $y_{best} \leftarrow y_1$;
17: **until** (termination criterion not reached);
18: **end**

ILS is based on the simple yet powerful idea of improving LS procedure by providing new starting solutions obtained from perturbations of a current solution. This procedure often leads to far better results than when using random restart. In general, four components have to be specified in order to apply ILS. These are a **GenerateInitialSolution** procedure that generates an initial solution s_0, a **Perturbation** procedure that modifies the current solution s leading to some intermediate solution s', a **LS** procedure that returns an improved solution s'' and a procedure **AcceptanceCriterion** that decides to which solution the next perturbation is applied. General framework of ILS is depicted in Algorithm 2.

Algorithm 2 Iterated local search (ILS) algorithm

Input: Solution s_0 from the population;

Output: an improved solution s.

1: **begin**
2: $s_0 \leftarrow$ an initial solution;
3: $s \leftarrow$ Apply LS with s_0;
4: **while** (termination criterion not met) **do**
5: $s' \leftarrow$ Perturbation(s, history);
6: $s'' \leftarrow$ Apply LS with s';
7: $s \leftarrow$ AcceptanceCriterion(s, s'', history);

8: **end**

The initial solution s_0 is provided to ILS as an argument from GA. Now, the LS algorithm with some neighborhood structures is applied to s_0 in order to find a better solution s. Once a better solution is found, perturbation is performed on s to obtain a new solution s'. After getting s', LS is applied again on it to obtain a new local optimal solution s''. The solution s'' replaces s' only if it satisfies the acceptance criterion. Here, history is corresponding to the search history. The ILS process continues until a stopping criterion is met. The main drawback of LS is that it gets trapped in local optima that are significantly worse than the global optimal. ILS escapes from local optima by applying perturbations to the current local optimum. Each time a perturbation is made throughout the ILS procedure, a new solution is created. The strength of a perturbation is referring as the number of solution components which are modified. In this problem, we implemented the four types of perturbation moves. Perturbation move P_1 chooses a different timeslot for a randomly chosen event; P_2 swaps the timeslots of two randomly chosen events; P_3 randomly select two timeslots and swaps all their events; and P_4 permutes three events in three distinct timeslots in one of the two possible ways other than the existing permutation of the three events. All random choices were taken according to a uniform distribution. Each different move is applied k times, where k is chosen from the set {1, 5, 10, 25, 50, 100}. This Perturbation is applied to the solution returned by the AcceptanceCriterion. The acceptance criterion forces the cost to decrease. In this problem, three different methods for accepting solutions are considered in AcceptanceCriterion. The first method, `Random Walk`, always accepts the new solution s'' returned by LS. The second method, `Accept if Better`, accepts the new solution s'' if it is better than s. This method leads to a first improvement descent in the space of the local optima. The third method, `SA`, accepts the new solution s'' if it is better than the current one. Otherwise s'' is accepted with a probability based on the evaluation function $g(s)$, but infeasible new solutions are never accepted when the current one is feasible. $g(s)$ is the number of hard-constraint violations if both s and s'' are infeasible, or the number of soft-constraint violations if they are both feasible. Two methods for calculating this probability were applied;

$$M_1 : P_1(s, s'') = e^{\frac{(g(s)-g(s''))}{T}} \text{ and } M_2 : P_2(s, s'') = e^{\frac{(g(s)-g(s''))}{T.g(s_{best})}}$$

where T is a parameter called temperature and s_{best} is the best solution found so far. The value of T is kept fixed during the run, and it is chosen from {0.01, 0.1, 1} for M_1 and {0.05, 0.025, 0.01} for M_2.

4. Experimental results

In this section, the performance of the proposed hybrid algorithm GAILS combining GA with ILS is investigated by comparing it with a number of existing state-of-the-art algorithms used for the solution of UCTPs. The performance measure used is in terms of fitness function values. All algorithms are programmed in C++ on GNU compiler GCC version 4.5.2 running on a 3.10 GHz PC. Datasets of problems instances are taken from the benchmark problems proposed in Rossi-Doria et al. (2003). These problem instances are divided into three small, medium and large categories. The population size, the mutation probability and the crossover probability of GAILS are taken as 10, 0.5, and 0.8 respectively.

The best resulting configurations of parameters for small problem instances are: type of Perturbation is P_1 with k = 1 and AcceptanceCriterion is M_2 with T = 0.025. For medium and large problem instances these parameters are: type of Perturbation is P_1 with k = 5 and AcceptanceCriterion is M_1 with T = 0.1. All small, medium and large problem instances are run independently for 200, 50 and 20 trials and the smallest fitness function value among them is taken as the best value for the solution of the problem instance. The maximum numbers of iterations for the LS are fixed by 200, 100000 and 100000 for small, medium and large problem instances. Also, each class of problem instances has been determined experimentally by given a specified time limit. For all the small problem instances it is fixed by 2 seconds. It is observed that GAILS is able to produce optimal solution with zero fitness function value for all the small problem instances in every independent run. All the medium problem instances are performed over time limit 900 seconds. In the similar manner, all the large problem instances are performed over time limit 9000 seconds. The best solution obtained in these time durations along with average solutions, standard deviations and exact time to achieve the optimum solution for all the problem instances are given in Table 1. Also, the fitness function values versus time taken by GAILS for all the small, medium and large problems instances are depicted by the graphs are given in Figures 1(a), 1(b) and 1(c). In order to show the effectiveness of the GAILS, we compare our obtained results with the results obtained by some state-of-the-art algorithms available in the literature. Table 2 shows the comparison of the best fitness function values obtained.

Terms *x* % Inf. and *NA* in Table 2 indicates a percentage of runs that failed to produce feasible solutions and not available solution. From the aforementioned experimental results, it can be seen that newly developed GAILS technique can help to minimize the objective function value and give better results for the consider UCTP compared to other state-of-the-art algorithms employed in the literature.

Table 1. Output of considered problem instances

Dataset	f_{min}	f_{max}	f_{avg}	Standard deviation	Time (in seconds)
small01	0	0	0	0	0.052003
small02	0	0	0	0	0.016
small03	0	0	0	0	0.008
small04	0	0	0	0	0.152010
small05	0	0	0	0	0.020002
med01	92	112	102.2333	5.4448	818.1391
med02	82	120	99.1333	9.7902	803.5142
med03	122	159	139.7667	12.284	857.3456
med04	73	106	90.3667	9.2393	641.1961
med05	89	128	109.9	11.9313	871.6945
hard01	585	708	635.35	39.2228	8392.2334
hard02	489	578	514.9	22.5851	8821.5228

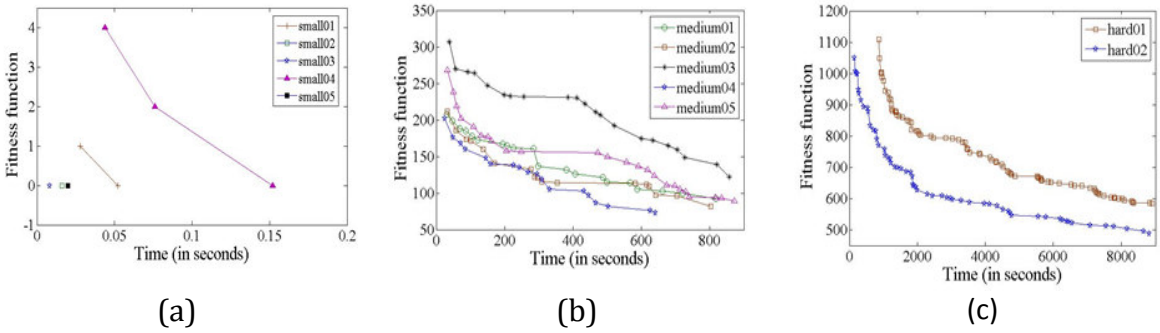

Figure 1. Fitness function values versus time for (a) small instances (b) medium instances (c) hard instances

Table 2. Comparison results of considered problem instances

Instance	GAILS	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
small01	0	0	0	0	2	0	0	0	0	0	0	0
small02	0	0	3	0	4	0	0	3	0	0	0	0
small03	0	0	0	0	2	0	0	0	0	0	0	0
small04	0	0	0	0	0	0	0	0	0	0	0	0
small05	0	0	0	0	4	0	0	0	0	0	0	0
med01	92	106	280	128	254	221	139	96	80	124	117	150
med02	82	107	188	136	258	147	92	96	105	117	121	179
med03	122	132	249	197	251	246	122	135	139	190	158	183
med04	73	72	247	112	321	165	98	79	88	132	124	140
med05	89	107	232	171	276	130	116	87	88	73	134	152
hard01	585	505	Inf.	896	1026	529	615	683	730	424	645	750
hard02	489	486	1014	685	NA	NA	NA	NA	NA	NA	NA	NA

A1: Badoni et al. (2014); A2: Rossi-Doria et al. (2003); A3: Rossi-Doria et al. (2003); A4: Abdullah & Turabieh (2008); A5: Abdullah et al. (2007); A6: Yang & Jat (2011); A7: Abdullah et al. (2012); A8: Mcmullan (2007); A9: Al-Betar et al. (2010); A10: Ayob & Jaradat 2009; A11: Ayob & Jaradat 2009

4. Conclusions

A hybrid algorithm GAILS combining GA with ILS is described for solving UCTPs efficiently. It is based on ILS using three types of neighborhood moves and four types of perturbation moves. This allows us to improve each of the generation produced by GA. It is shown that in all small problem instances, GAILS gives optimal solutions with zero fitness function values within 0.2 seconds. Comparisons were carried out in order to show the effectiveness of our proposed algorithm over the other state-of-the-art algorithms available in the literature. It is observed that our method gives best fitness function values for medium02 and medium03 problem instances and second best fitness function value for medium01 and medium04 problem instances. This establishes that GAILS is an improved algorithm in comparison to GA and ILS algorithms.

References

Abdullah, S., Burke, E.K. & McCollum, B. (2007), "A hybrid evolutionary approach to the university course timetabling problem", IEEE Congress on Evolutionary Computation, 2007 (CEC'07), IEEE, 1764–1768.

Abdullah, S. & Turabieh, H. (2008), "Generating university course timetable using genetic algorithms and local search", Third International Conference on Convergence and Hybrid Information Technology, 2008 (ICCIT'08), IEEE, 254–260.

Abdullah, S., Turabieh, H., McCollum, B. & McMullan, P. (2012), "A hybrid metaheuristic approach to the university course timetabling problem", Journal of Heuristics 18, Springer, 1–23.

Al-Betar, M.A., Khader, A.T. & Liao, I.Y. (2010), "A harmony search with multi-pitch adjusting rate for the university course timetabling", Recent Advances In Harmony Search Algorithm, Springer, 147–161.

Ayob, M. & Jaradat, G. (2009), "Hybrid ant colony systems for course timetabling problems", 2nd Conference on Data Mining and Optimization (DMO'09), IEEE, 120–126.

Badoni, R.P., Gupta, D. & Mishra, P. (2014), "A new hybrid algorithm for university course timetabling problem using events based on groupings of students", Computers & Industrial Engineering 78, Elsevier, 12–25.

John, H. (1992), "Adaptation in natural and artificial systems", MIT Press, Cambridge, MA.

Mcmullan, P. (2007), "An extended implementation of the great deluge algorithm for course timetabling", Computational Science 2007 (ICCS'07), Springer, 538–545.

Rossi-Doria, O., Sampels, M., Birattari, M., Chiarandini, M., Dorigo, M., Gambardella, L.M., Knowles, J., Manfrin, M., Mastrolilli, M., Paechter, B., Paquete & L., Stützle, T. (2003), "A comparison of the performance of different metaheuristics on the timetabling problem", Practice and Theory of Automated Timetabling IV, Springer, 329–351.

Turabieh, H. & Abdullah, S. (2009), "Incorporating tabu search into memetic approach for enrolment-based course timetabling problems", 2nd Conference on Data Mining and Optimization, 2009 (DMO'09), IEEE, 115–119.

Wijaya, T. & Manurung, R. (2009), "Solving university timetabling as a constraint satisfaction problem with genetic algorithm", Proceedings of the International Conference on Advanced Computer Science and Information Systems, 2009 (ICACSIS'09).

Yang, S. & Jat, S.N. (2011), "Genetic algorithms with guided and local search strategies for university course timetabling", Systems, Man, and Cybernetics, Part C: Applications and Reviews 41, IEEE Transactions, 93–106.

Biographical notes

Rakesh P. Badoni received his M.Sc. degree from Hemwati Nandan Bahuguna Garhwal University in 2005 and M.Tech. degree from IIT Kharagpur, India in 2011. Currently, he is working as a Research Scholar in the Department of Mathematics, IIT Kharagpur. His area of research is on theoretical computer science.

D.K. Gupta received his Ph.D. in Science from IIT Kharagpur, India in 1985. Later, he joined there as a faculty member in the Department of Mathematics. Currently, he is working as a Senior Professor. His research interests focus on theoretical computer science, distributed databases, constraint satisfaction problems and numerical analysis.

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar