2012 4th Conference on Data Mining and Optimization (DMO)
02-04 September 2012, Langkawi, Malaysia

A Differential Evolution Algorithm for the University Course Timetabling Problem

Khalid Shaker
Department of Software Engineering, Faculty of Computer Science and Information Technology, Universiti Malaya, 50603 Kuala Lumpur, Malaysia
khalidsh@um.edu.my

Salwani Abdullah and Arwa Hatem
Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia
{salwani, arwa}@ftsm.ukm.my

Abstract—**The University course timetabling problem is known as a NP-hard problem. It is a complex problem wherein the problem size can become huge due to limited resources (e.g. amount of rooms, their capacities and number availability of lecturers) and the requirements for these resources. The university course timetabling problem involves assigning a given number of events to a limited number of timeslots and rooms under a given set of constraints; the objective is to satisfy the hard constraints and minimize the violation of soft constraints. In this paper, a Differential Evolution (DE) algorithm is proposed. DE algorithm relies on the mutation operation to reduce the convergence time while reducing the penalty cost of solution. The proposed algorithm is tested over eleven benchmark datasets (representing one large, five medium and five small problems). Experimental results show that our approach is able to generate competitive results when compared with previous available approaches. Possible extensions upon this simple approach are also discussed.**

Keywords- differential evolution; course timetabling

I. Introduction

The CTTP is expressed as the minimization of the objective function $f(x)$: $x=(x_1,\ldots,x_n)^t$, where x is the initial feasible solution to which neighbourhood structures are applied in order to generate new feasible solutions. The objective function, $f(x)$, is minimized by an optimum solution vector $x^*=(x_1,\ldots,x_n)^t$ where $f(x^*) < f(x)$ for all x belonging to a feasible domain. Existing meta-heuristic optimization algorithms designed to solve the CTTP can be categorized as single-solution and population-based approaches. Surveys on meta-heuristic techniques are abundant in the literature (a typical recent review can be found in [1]). Examples of single-solution approaches are: tabu search [3, 4], graph coloring heuristics [4], great deluge [5.6.7], simulated annealing [8, 9], and variable neighbourhood search [10]. Population-based approaches include: genetic algorithms [11, 12], ant colony systems [13], memetic algorithms, evolutionary algorithms [14, 15], etc. Socha et al. [13] employed a local search and ant based algorithms, tested on eleven problems. Kostuch and Socha [16] investigated a statistical model in predicting the difficulty of timetabling problems particularly on the competition datasets. In 2007, Abdullah et al. [17] developed an iterative improvement algorithm with composite neighbourhood structures and later combined this algorithm with a mutation operator. McMullan [7] applied a two phased approach utilizing an adaptive construction heuristic and an extended version of the Great Deluge Algorithm. Landa-Silva and Obit employed a nonlinear great and deluge on the same instances [6]. Interested readers are referred to Lewis [1] for a comprehensive survey of the university timetabling approaches in recent years. In addition, other related papers on Enrolment-Based course timetabling problems can be found in Jat and Yang [18], and Pongcharoen et al. [19].

II. Problem Description

The enrolment-based course timetabling problem considered in this work was initially defined by the Metaheuristics Network[1]. This problem was discussed as an assignment of lecture events to timeslots and rooms according to a variety of hard and soft constraints. The problem description that is employed in this paper is adapted from the description presented in [13]. The problem involves scheduling 100-400 courses into a timetable with 45 timeslots corresponding to 5 days of 9 hours each, whilst satisfying room features and room capacity constraints. They are divided into three categories: small, medium and large. We deal with 11 instances: 5 small, 5 medium and 1 large. The characteristics which define the categories are given in Table I.

TABLE I. THE DESCRIPTION OF THE ENROLMENT-BASED INSTANCE

Category	*small*	*Medium*	*large*
Number of courses	100	400	400
Number of rooms	5	10	10
Number of features	5	5	10
Number of students	80	200	400
Maximum courses per student	20	20	20
Maximum student per courses	20	50	100
Approximation features per room	3	3	5
Percentage feature use	70	80	90

The problem has:

- A set of N courses, $e = \{e_1,\ldots,e_N\}$
- 45 timeslots
- A set of R rooms
- A set of F room features

[1] http://www.metaheuristics.net

- A set of M students.

This problem includes four hard constraints and three soft constraints as follows:

Hard constraints:

- *Event conflict* i.e. no student can be assigned to more than one course at the same time (coded as H_1).
- *Room features* i.e. the room should satisfy the features required by the event (coded as H_2)
- *Room capacity* i.e. the number of students attending the event should be less than or equal to the capacity of the room (coded as H_3).
- *Room occupancy* i.e. no more than one event is allowed at a timeslot in each room (coded as H_4).

Soft constraints:

- *Event in the last timeslot* i.e. a student shall not have to sit a course that is scheduled in the last timeslot of the day (coded as S_1)
- *Two consecutive events* i.e. a student shall not have more than 2 consecutive events (coded as S_2).
- *One event a day* i.e. a student shall not have to sit a single course on a day (coded as S_3).

Hard constraints act an inviolable requirement. A timetable which meets the hard constraints is recognised as a *feasible* solution. The main objective is to minimise the violation of the soft constraints in a *feasible* solution that later represents the quality of the obtained solution. A solution consists of an ordered list of length $|N|$ where the position corresponds to the events i.e. position i corresponds to event e_i for $i = 1,\ldots,|N|$. The values for each position are a number in between 0 to 44 corresponding to the timeslot index, and 0 to $|R\text{-}1|$ correspond to the room index. For example, a timeslot vector is given as (0,17,30,…,10) and a room vector is given as (4,3,0,…,3) means that event e_1 is scheduled in timeslot 0 in room 4. Event e_2 is scheduled in timeslot 17 in room 3 and finally event $e_{|N|}$ is scheduled in timeslot 10 in room 3.

The objective function for the problem is defined in the formula below.

$$min \sum s_1 + s_2 + s_3 \qquad (1)$$

where S_1, S_2 and S_3 represent the relevant soft constraints.

III. The Algorithm

The proposed algorithm consists of two phases i.e. build a feasible initial population using a constructive heuristic; and an improvement algorithm with an aim to optimise the violation of the soft constraints while maintaining the feasibility of the solutions.

A. Neighbourhood Structure

The different neighbourhood structures and their explanation can be outlined as follows:

N_1: Choose a single course at random and move to a feasible timeslot that can generate the lowest penalty cost.

N_2: Select two courses at random from the same room (the room is randomly selected) and swap timeslots.

B. Constructive Heuristic

A least saturation degree is used to generate initial solutions (the population) which start with an empty timetable [7]. The events with less rooms available and more likely to be difficult to be scheduled will be attempted first without taking into consideration the violation of any soft constraints, until the hard constraints are met. This process is carried out in the first phase. If a feasible solution is found, the algorithm stops. Otherwise, phase 2 is executed. In the second phase, neighbourhood moves (N1 and/or N2) are applied with the aim to change infeasible solution to feasible one. N1 is applied for a certain number of iterations. If a feasible solution is met, then the algorithm stops. Otherwise the algorithm continues by applying a N2 neighbourhood structure for a certain number of iterations. We have no proof that this constructive heuristic is guaranteed to find a feasible solution for a given instance. However, in this experiment, the solutions were made feasible before the improvement algorithm is applied.

C. Improvement Algorithm: Differential Evolution Algorithm

DE is a basic algorithm of the population that employed crossover, mutation and selection operators as in genetic algorithms. The main difference in obtaining better solutions is that genetic algorithms rely on the crossover operation, while in the DE algorithm it is based on the mutation operation. The algorithm uses mutation operation as a search mechanism and selection operation to direct the search towards the potential regions in the search space. This population is then improved by applying mutation, crossover and selection operators. The main steps of the differential evolution algorithm are given in Fig. 1.

```
Initialization
Evaluation
do while (termination criteria are met)
  Mutation
  Crossover
  Evaluation
  Selection
end do while
```

Figure 1. Differential Evolution Algorithm

The algorithm starts with the randomly selecting two parents (p_1, p_2) from the population. The mutation operator carried out on selected parent timetables (p_1, p_2) as one of the neighbourhood structures (as listed in Section III) selected randomly will be applied to both parent timetables (p_1, p_2) separately, to generate a new parent timeslots (p_1*, p_2*). Then, a crossover operator is defined: taking the two parent timetables (p_1*, p_2*) and exchanging two timeslots selected randomly between p_1*and p_2*, and allocating rooms to events in each time slot. Two new offspring will be generated. These offspring solutions are called *child*$_1$ and *child*$_2$. Then the best among the new offspring will be selected by using the evaluation and selection operator. The quality of the new offspring ($f(child_1)$, $f(child_2)$) is calculated and compared with the quality of the best solution, $f(BestSol)$. If there is an improvement, the best new solution among $child_1$ and $child_2$ will be accepted to be in the population after removing the worst solution from population. The processes will be repeated and the algorithm stops when the maximum number of iterations is reached or the penalty cost is zero.

1. *Chromosome Representation*

A simple chromosome representation can improve the crossover and mutation operations [19]. Fig. 2 shows an example of the chromosome representation that is composed of a string of genes. Each chromosome represents a feasible timetable, where the gene represents timeslot t_i, room r_i, and course c_i. For example, courses c_1, c_5, c_7, c_{12} are scheduled at timeslot t_1 in the rooms r_1, r_2, r_3 and r_4, respectively.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	c_1	c_{10}	.	.	.	.	.	.	c_3
r_2	c_5	c_{11}	c_8	.	.	.	.	.	c_6
r_3	c_7	.	c_{13}	.	.	.	.	.	c_9
r_4	c_{12}	.	c_{16}	.	.	.	.	.	.

Figure 2. Representation of chromosome for course timetabling problems

2. *Mutation and Crossover Operations*

The mutation is considered as a main operation for DE algorithm, as it relies on mutation operation (Storn [20]). Random selections of neighborhood structures (as listed in Section III) are used in a mutation operation which is based on a mutation rate.

The crossover operation is illustrated as in Fig. 3 as proposed by Abdullah et al. [11].

Parent1

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	c_{18}		.	c_4	c_{10}	c_{17}	c_2	c_{19}	c_{13}
r_2	c_{14}	c_6	c_5	c_{12}	c_1	.	.	.	c_{15}
r_3	c_{20}	c_7	c_3	.	.	c_9	c_{16}	.	.
r_4	c_{21}	.	c_8	.	c_{11}	.	.	c_{22}	.

Parent2

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	c_{19}	c_4	.	.	c_{20}	c_{17}	c_2	c_8	c_5
r_2	c_{13}	c_{16}	.	c_{21}	c_{14}	.	.	.	c_{16}
r_3	.	.	c_3	c_6	.	c_{12}	.	c_7	.
r_4	c_9	c_{10}	.	.	c_1	c_{18}	.	c_{22}	c_{11}

Offspring1

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	c_{18}	c_8	.	c_4	c_{10}	c_{17}	c_2	c_{19}	c_{13}
r_2	c_{14}	c_6	c_5	c_{12}	c_1	.	.	.	c_{15}
r_3	c_{20}	c_7	c_3	.	.	c_9	c_{16}	.	.
r_4	c_{21}	c_{22}	~~c_8~~	.	c_{11}	.	.	~~c_{22}~~	.

Offspring2

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
r_1	c_{19}	c_4	.	.	c_{20}	c_{17}	c_2	c_8	c_5
r_2	c_{13}	c_{16}	.	c_{21}	c_{14}	.	.	c_6	c_{16}
r_3	.	.	c_3	~~c_6~~	.	c_{12}	.	c_7	.
r_4	c_9	c_{10}	.	.	c_1	c_{18}	.	c_{22}	c_{11}

Figure 3. Crossover operation

The crossover operator exchanges the shaded timeslots between p_1 and p_2 to form the offspring. The shaded timeslots, i.e. t_2 and t_8 are randomly selected. In this case, the feasibility of the solution is likely be violated. In order to maintain the feasibility of the offspring, two conditions are considered:

1) Course can be moved only if the corresponding timeslots are empty. For example, c_6 can be moved from timeslot t_2 in Parent1 to timeslot t_8 in Parent2.
2) No conflicts occur between moved course and scheduled courses. For example, in Offspring1, there should be no conflict between moved course c_8 from t_8 (Parent2) and scheduled courses c_6, c_7 in t_2 (Parent1).

IV. Experimental Results

The algorithm is coded using Matlab under Windows XP. All experiments were run on an Intel CoreTM with a 1.86GHz processor. We evaluate our results on the instances taken from Socha et al [13] and which are available at http://iridia.ulb.ac.be/~msampels/tt.data/. Table II shows the parameters setting for the proposed algorithm after some preliminary experiments and almost similar with the papers in the literature (e.g. Yang and Jat [18]). We ran the experiments for 200000 generations with 11 test-runs to obtain an average value.

TABLE II. Parameters Setting of DE Algorithm

Parameters	Enrolment-based CTTP
Generations number	200000
Population size	50
Crossover rate	0.8
Mutation rate	0.5

The best results out of 11 runs obtained are presented in Table III that shows the comparison of the approach in this work against other available approaches reported in the literature. Where [11] used genetic algorithm and local search, [15] used hybrid evolutionary approach, [21] used electromagnetic-like mechanism, [22] used Harmony search, [7] used an extended great deluge, [6] used a nonlinear great deluge, [5] used an incorporating great deluge with kempe chain, and [8] used Dual Sequence Simulated Annealing with Round-Robin Approach.

TABLE III. Results Comparison

Dataset	Proposed method	[11]	[15]	[21]	[22]	[7]	[6]	[5]	[8]
small1	**0**	**0**	**0**	**0**	**0**	**0**	3	**0**	**0**
small2	**0**	**0**	**0**	**0**	**0**	**0**	4	**0**	**0**
small3	**0**	**0**	**0**	**0**	**0**	**0**	6	**0**	**0**
small4	**0**	**0**	**0**	**0**	**0**	**0**	6	**0**	**0**
small5	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
medium1	160	175	221	96	124	**80**	140	98	93
medium2	**81**	197	147	**96**	117	105	130	113	98
medium3	149	216	246	135	190	139	189	**123**	149
medium4	113	149	165	**79**	132	88	112	100	103
medium5	143	190	130	87	**73**	88	141	135	98
large	735	912	529	683	**424**	730	876	610	680

Note that the best results are presented in bold. It can be seen that in general, the proposed approach here is better than other approaches reported in the literature as it is able to generate best solution on *medium2* dataset.

It can be clearly seen that the DE approach performs well for the *small* datasets where the algorithm is able to obtain the optimal solutions with zero penalty, because the *small* datasets might have more feasible solution points in the search space compared to the *medium* and *large* datasets.

Figures 4 (a) and (b) illustrate the effectiveness of the algorithm when it is exploring the search space on *small5and medium5*. The x-axis represents the number of iterations, while the y-axis represents the penalty cost.

The figures show that the algorithm is able to produce faster convergence as the DE algorithm significantly reduces the search space by only considering feasible solution spaces.

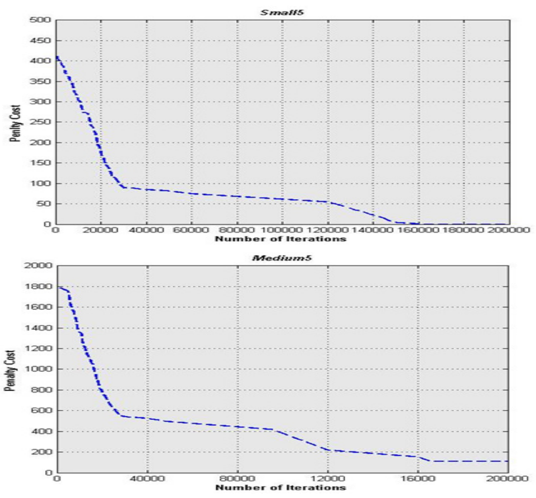

Figure 4. Convergences results of (a) small5 (b) medium5 datasets

In most cases *small*, the way the algorithm explores the search space clearly indicates that further improvement is achieved. In addition, less complexity of *small* datasets allows the technique to explore a number of non-improvement moves and achieve a zero evaluation. The algorithm is also effective for the m*edium* and *large* datasets, in the graphs clearly indicating that a further reduction of more than 60% in the cost evaluation can be achieved. We believe that the algorithm can fast when it relies on the mutation operation rather than the crossover operation.

V. Conclusion and Future Work

This paper presents an effective Differential Evolution Algorithm (DE) for the university course timetabling problems, showing that evolutionary computation can deal successfully with the problem. The use of the effective Differential Evolution Algorithm (DE) is an important element of its high performance. There are two advantages from using DE; first, fast convergence, and second, a few control parameters are use. Preliminary comparisons indicate that this algorithm is competitive with other approaches in the literature. Future research will be aimed to test this algorithm on the International Timetabling Competition datasets (ITC2007).

References

[1] R Lewis. A survey of metaheuristic-based techniques for university timetabling problems, OR Spectrum 30 (1), 167-190 (2008).

[2] EK Burke, G Kendall and E Soubeiga, A tabu search hyperheuristic for timetabling and rostering. Journal of Heuristics 9(6), 451-470 (2003).

[3] R Alvarez-Valdes, E Crespo and JM Tamarit. Design and implementation of a course scheduling systems using tabu search: Production, manufacturing and logistics. European Journal of Operational Research, 137, pp 512-523 (2002).

[4] EK Burke, B McCollum, A Meisels, S Petrovic and R Qu, A Graph-Based Hyper Heuristic for Educational Timetabling Problems, European Journal of Operational Research 176(1), 177-192 (2007).

[5] S Abdullah, K Shaker, B McCollum & P McMullan. Incorporating great deluge with Kempe chain neighbourhood structure for the enrolment-based course timetabling problem. RSKT 2010, In LNAI: Vol. 6401. Springer, pp 70–77, 2010.

[6] D Landa-Silva and JH Obit. Great Deluge with Nonlinear Decay Rate for Solving Course Timetabling Problems. Proceedings of the 2008 IEEE Conference on Intelligent Systems (IS 2008), IEEE Press, 8.11-8.18 (2008)

[7] P McMullan. An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, Lecture Notes in Computer Science, Springer, Vol 4487, pp538-545 (2007)

[8] S Abdullah, K Shaker, B McCollum, P McMullan. Dual Sequence Simulated Annealing with Round-Robin Approach for University Course Timetabling. EVOCOP 2010, LNCS: 6022, Springer-Berlin /Heidelberg, pp 1–10, 2010.

[9] R Bai, EK. Burke, G Kendall and B McCullum. "A Simulated Annealing Hyper-heuristic for University Course Timetabling Problem". (Abstract) PATAT '06, Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, ISBN 80-210-3726-1, pp345-350 (2006).

[10] S Abdullah, EK Burke and B McCollum, An investigation of variable neighbourhood search for university course timetabling. The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), 413-427 (2005).

[11] S Abdullah and H Turabieh, Generating university course timetable using genetic algorithms and local search. The Third 2008 International Conference on Convergence and Hybrid Information Technology ICCIT, vol. I, 254-260 (2008).

[12] R Lewis and B Paechter. New crossover operators for timetabling with evolutionary algorithms. Proceedings of the 5th International Conference on Recent Advances in Soft Computing (ed. Lotfi), UK, December 16th-18th, pp 189-194 (2004).

[13] K Socha, J Knowles and M Samples, A max-min ant system for the university course timetabling problem. Proceedings of the 3rd International Workshop on Ant Algorithms (ANTS 2002), Springer Lecture Notes in Computer Science Volume 2463, 1-13 (2002).

[14] Jat, S. N., & Yang, S. (2008). A memetic algorithm for the university course timetabling problem. Proc. of the 20th IEEE international conference. tools with artificial intelligent. (pp. 427–433).

[15] S Abdullah, EK Burke and B McCollum, A Hybrid Evolutionary Approach to the University Course Timetabling Problem. IEEE Congres on Evolutionary Computation, ISBN: 1-4244-1340-0, 1764-1768 (2007).

[16] P Kostuch and K Socha, Hardness Prediction for the University Course Timetabling Problem, Proceedings of the Evolutionary Computation in Combinatorial Optimization (EvoCOP 2004), Coimbra, Portugal, April 5-7, 2004, Springer Lecture Notes in Computer Science Volume 3004, 135-144 (2004).

[17] S Abdullah, EK Burke and B McCollum, Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for University Course Timetabling. In: Metaheuristics Progress in Complex Systems Optimization, Springer, 153-169 (2007).

[18] Jat, S. N. and Yang, S. 2010. A hybrid genetic algorithm and tabu search approach for post enrolment course timetabling. Journal of Scheduling pp.1-21.

[19] P. Pongcharoen, W. Promtet, P. Yenradee and C. Hicks, "Stochastic optimisation timetabling tool for university course scheduling," International Journal of Production Economics 112 (2): 903 – 918, special Section on RFID: Technology, Applications, and Impact on Business Operations, 2008.

[20] R Storn, "Differential Evolution, A Simple and Efficient Heuristic Strategy for Global Optimization over Continuous Spaces" Journal of Global Optimization, Vol. 11, Dordrecht, pp. 341-359, 1997.

[21] H Turabieh,S Abdullah, B McCollum. Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem. RSKT 2009. LNCS, vol. 5589, pp. 497–504. Springer, Heidelberg, 2009.

[22] M Al-Betar, A. Khader and I. Yi Liao, "A Harmony Search with Multi-pitch Adjusting Rate for the University Course Timetabling," In: Z.W. Geem: Recent Advances in Harmony Search Algorithm, SCI 270, pp. 147–161. Springer, Heidelberg, 2010.