International Journal of the Physical Sciences Vol. 6(6), pp. 1452-1462, 18 March, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.210
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Investigating a round robin strategy over multi algorithms in optimising the quality of university course timetables

Salwani Abdullah[1], Khalid Shaker[1]* and Hothefa Shaker[2]

[1]Data Mining and Optimisation Research Group (DMO), Center for Artificial Intelligence Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia.
[2]Department of Computer Science, Universiti Tenaga Nasional, Jalan Ikram-Uniten, 43000 Kajang, Selangor, Malaysia.

Accepted 3 March, 2011

This paper tackles university course timetabling problems (UCTP) to find a (near) optimal solution (timetable) while satisfying hard constraints (essential requirements) and minimizing as much as possible the violations of the soft constrains (desirable requirements). In this study, we apply three algorithms, that is, Great Deluge, Simulated Annealing and Hill Climbing where the Round Robin algorithm is used as control strategy in choosing the algorithm to be employed at the current stage. The performance of the approach is tested with over two sets of benchmark datasets, that is, enrolment-based course timetabling and curriculum-based course timetabling (UD1) in comparison with a set of state-of-the-art methods from the literature. The experimental results show that the proposed approach is able to produce competitive results for the test UCTPs.

Key words: University course timetabling problem, round robin, great deluge, simulated annealing, hill climbing algorithm.

INTRODUCTION

In university course timetabling problems (UCTP), a set of courses are scheduled into a given number of rooms and timeslots across a period of time. This usually takes place within a week and the resultant timetable replicated for as many weeks as the courses run. Also, students and teachers are assigned to courses so that the teaching delivery activities can take place. The course timetabling problem is subject to a variety of hard and soft constraints. Hard constraints need to be satisfied in order to produce a *feasible* solution. In contrast, violations of soft constraints are possible but must be minimized in order to achieve the best possible quality of the solution.

Over the past four decades, researchers have proposed various approaches to solve university course timetabling problems by using single-based methods, metaheuristic methods (e.g. tabu search (Glover 1997), simulated annealing, and great deluge), population-based methods (e.g. genetic algorithms and ant colony optimization), and hybrid/hyper-heuristic approaches, etc. Interested readers are referred to Lewis (2008) for a comprehensive survey of the university timetabling approaches in recent years and (Ross and Corne 1995) for a comparison between a genetic algorithms, simulated annealing and stochastic hill climbing. The following provides an overview of techniques which have been used to find solutions to various formulations of the course timetabling problem in the past. Socha et al. (2002) employed a local search and ant based algorithms, tested on the eleven problems produced by Paechter's[1] course timetabling test instance generator (these instances are used to evaluate the method described in this paper). Burke et al. (2003a) introduced a tabu-search hyper-heuristic which was also tested on a nurse rostering problem.

*Corresponding author. E-mail: khalid@ftsm.ukm.my.

[1] http://www.dcs.napier.ac.uk/~benp/

The great deluge algorithm was employed by Burke et al. (2003b). Di Gaspero and Schaerf (2003) applied a multi-neighbourhood search approach, tested on the same instances. Lewis and Paechter (2004) designed several crossover operators, tested on a further twenty instances generated by Paechter's generator, used in the first competition in 2002 (http://www.idsia.ch/Files/ttcomp2002).

Kostuch and Socha (2004) investigated a statistical model in predicting the difficulty of timetabling problems particularly on the competition datasets. Kate et al. (2003) considered the discrete Hopfield neural networks for solving school timetabling problems. Kostuch (2005) presented a three phase approach employing simulated annealing and achieving 13 of the best results from the 20 competition instances. A variable neighbourhood search with a fixed tabu list was employed by Abdullah et al. (2005). Asmuni et al. (2005) applied a fuzzy multiple heuristic ordering on the eleven standard benchmark datasets.

Abdullah et al. (2007) developed an iterative improvement algorithm with composite neighbourhood structures and later combined this algorithm with a mutation operator. McMullan (2007) applied a two-phased approach utilizing an adaptive construction heuristic and an extended version of the Great Deluge algorithm. Abdullah and Turabieh (2008) employed a genetic and local search approach on the eleven benchmark course data sets. Landa-Silva and Obit (2008) employed a non linear great deluge on the same instances.

A great deluge with kempe chain neighbourhood structure was employed by Abdullah et al. (2010b) to solve university course timetabling. A gap between theory and practice in the area of university timetabling by McCollum (2007) and other related papers on Enrolment-Based course timetabling problem are Burke et al. (2007), McCollum et al. (2010), Chiarandini et al. (2006) Lu and Hao (2008), and Dimopoulou and Miliotis (2004).

Müller (2008) applied a constraint-based solver approach to the curriculum-based course timetabling problems in the 2nd International Timetabling Competition (Track 1 and 3) as introduced by Di Gaspero et al. (2007) and achieved first place in this competition. Lu and Hao (2010) applied a hybrid heuristic algorithm called adaptive tabu search to the same instances.

Burke et al. (2009) introduced a new solver based on a hybrid meta-heuristic to tackle scheduling problems. They applied it first on the 2nd International Timetabling Competition (Track 3) and was able to achieve good solutions within a practical timeframe. Sadaf and Shengxiang (2010) proposed a hybrid approach to solve post enrolment course timetabling problem in two phases. First phase a genetic algorithm applied to guide the search as it uses a data structure to store useful information from previous good individuals. Local search algorithms are used to enhance the individuals and tabu search algorithm applied on the best solution obtained from the first phase to improve the optimality of the solution. Other papers that tackle curriculum-based course timetabling problems can be found (Clark et al., 2008; De Cesco et al., 2008; Geiger, 2008; Lach and Lubbecke, 2008).

PROBLEM DESCRIPTION

In this work, two sets of problems are considered, that is, enrolment-based course timetabling problem and curriculum-based course timetabling problem. The description of the problem is discussed as follows.

Enrolment-based course timetabling problem

The problem description that is employed for the first problem here is adapted from the description presented in Socha et al. (2002) who present the following hard and soft constraints:

HC_1. *No student can be assigned to more than one course at the same time.*
HC_2. *The room should satisfy the features required by the course.*
HC_3. *The number of students attending the course should be less than or equal to the capacity of the room.*
HC_4. *Not more than one course is allowed to be assigned to a timeslot in each room.*

Soft constraints that are equally penalized are as follows:

SC_1. *A student has a course scheduled in the last timeslot of the day.*
SC_2. *A student has more than 2 consecutive courses.*
SC_3. *A student has a single course in a day.*

The problem has:

1. A set of n courses, $E = \{e_0, e_1,\ldots,e_{n-1}\}$
2. 45 timeslots, $T = \{t_0, t_1,\ldots,t_{44}\}$
3. A set of m rooms, $R = \{r_0, r_1,\ldots,r_{m-1}\}$
4. A set of q room features, $F = \{f_0,\ldots, f_{q-1}\}$
5. A set of v students $S = \{s_0, s_1,\ldots,s_{v-1}\}$.

The objective of this problem is to satisfy the hard constraints and to minimise the violation of the soft constraints.

The formula represents the objective function for this problem is given as below:

$$min \sum_{i=1}^{v} f(SC_1) + f(SC_2) + f(SC_3)$$

Curriculum-based course timetabling problem

This problem is taken from the international timetabling competition (ITC2007) that consists of the weekly scheduling of lectures for several university courses within a given number of rooms and time periods, where conflicts between courses are set according to the curricula of the university. The problem consists of the following basic entities:

Days, timeslots and periods

The timetable consists of a number of teaching days in the week, usually 5 or 6 days according to the university system. Each day is split into a fixed number of timeslots, the same for all days. A period is the combination of day and timeslot. The total number of scheduling periods is the product of the number of days and number of timeslots.

Courses and teachers

Each course consists of a fixed number of lectures to be scheduled in distinct periods, attended by a given number of students, and taught by a teacher. For each course, there is a minimum spread (in terms of days) for the lectures of the course, and there are some periods in which the course cannot be scheduled.

Rooms

Each room has a capacity (number of available seats) and location (an integer value representing a separate building). Some rooms are not suitable for some courses due to a lack of required equipment.

Curricula

A curriculum is any pair of courses that have common students. Conflicts between courses and other soft constraints are built according to the curricula published by the university.

The aim is to assign of all lectures of each course to a period (a pair composed of a day and a timeslot) and a room in order to achieve a solution, taking into account hard and soft constraints violations. All details, updates and news about the problem can be obtained via the website (http://tabu.diegm.uniud.it/ctt/index.php).

The following hard and soft constraints are presented:

Hard constraints:

1. Lectures: All lectures of a course must be scheduled, and they must be assigned to distinct periods.
2. Conflicts: Lectures of courses in the same curriculum or taught by the same teacher must all be scheduled in different periods.
3. Room occupancy*:* Two lectures cannot take place in the same room in the same period.
4. Availability: The teacher of the course must be available to teach that course at a given period; otherwise no lecture of the course can be scheduled at that period.

Soft constraints:

1. Room capacity. The number of students attending the course should be less than or equal to the capacity of the room.
2. Minimum working days: The lectures of each course must be spread into the given minimum number of days.
3. Isolated lectures: Lectures belonging to a curriculum should be in consecutive periods.
3. Room stability: All lectures of a course *should* be given in the same room.

THE PROPOSED ALGORITHM

The search algorithm consists of two stages. The first stage, that is, a constructive stage is concerned to produce an initial solution where a least saturation degree heuristic and largest degree heuristic are used to generate initial solutions for enrolment-based and curriculum-based course timetabling problems, respectively. The second stage, that is, an improvement stage aims to optimise the quality of the generated timetables by minimising the violations of the soft constraint violations.

Constructive heuristic

Enrolment-based course timetabling problem (EBCTT)

In this problem, a least saturation degree heuristic is used to generate initial solution starting with an empty timetable (McMullan, 2007). Firstly, the events with fewer rooms available and more likely to be difficult to schedule will be attempted to be scheduled first, without taking into consideration the violation of any soft constraints. If a feasible solution is found, the algorithm terminates. Otherwise, neighbourhood moves (coded as N_1 and N_2) are applied with an aim to achieve feasibility. N_1 is applied for a certain number of iterations (set to 500, from experimentation). If a feasible solution is met, then the algorithm stops. Otherwise, the algorithm continues by applying N_2 neighbourhood move for a certain number of iterations. Across all instances tested, solutions were made feasible before the improvement algorithm was applied. The description of the neighbourhood moves can be found in “Neighbourhood moves” in this paper.

Curriculum-based course timetabling problem (CBCTT)

In this problem, a larger degree heuristic is employed that starts with an empty timetable (Gaspero and Schaerf, 2004). The degree of an event is a count of the number of other events which conflict, in the sense that students are enrolled in both events. This heuristic orders events in terms of those with the highest degree first (Landa-Silva and Obit, 2008). The events with highest degree of conflict will be attempted first without taking into consideration the violation of any soft constraints, until the hard constraints are met. All events are scheduled by randomly selecting the timeslot and the room that satisfies the hard constraints. Some events cannot be scheduled to a specific room; in this case, they will be inserted in any randomly selected room. If all hard constraints are met, then the feasible solution is found and the algorithm terminates. Otherwise, neighbourhood moves as the one applied for the enrolment-based course timetabling problem is executed.

IMPROVEMENT ALGORITHM

During the optimisation process, the neighbourhood moves are applied in all three algorithms, that is, Hill Climbing, Great Deluge and Simulated Annealing. Hard constraints are never violated during the timetabling process. The general pseudo code of the improvement algorithm is given in Figure 1.

The RR algorithm is employed to control the applying of the three algorithms, which are ordered in sequence. In this work, the algorithms are ordered as ALG_1 (Hill Climbing), ALG_2 (Great Deluge) and ALG_3 (Simulated Annealing). A time quantum is assigned for each algorithm in equal portions, in a circular order. The algorithm is dispatched in a FIFO manner at a given quantum denoted as q_time which is set to 15 min as we did experiment with several time quantum, namely, 5, 10, 15, 20, 30 min. Finally we choose a fixed value (q_time = 15 min) is used for medium and large data sets, and 10 s for *small* datasets (for the case of

```
Set the initial solution Sol by employing a constructive heuristic;
Calculate initial cost function f(Sol);
Set best solution Solbest ← Sol;
Set quantum time, q_time;
Set initial value to counter_qtime;

 do while (not termination criteria)
        Set a sequence algorithms in a queue which is ordered as
        ALGi where i ∈ {1,…,K} and K = 3;
         do while (q_time not met )
         Select an algorithm ALGi in the queue where i ∈ {1,…,K};
    A:  Apply ALGi on current solution, Sol to generate new
         solution, Sol*;
         if there is an improvement on the quality of the solution then
            update Solbest, Sol;
            repeat label A
        else
            reset ALGi parameters;
            insert ALGi into the queue;
```

Figure 1. The pseudo code for the improvement algorithm.

Table 1. Parameter setting of the algorithm.

Parameter	**Hill climbing**		**Great Deluge**		**Simulated annealing**	
	EBCTT	**CBCTT**	**EBCTT**	**CBCTT**	**EBCTT**	**CBCTT**
Number of liter	10000	-	10000	-	10000	-
Execution time (Seconds)	-	600	-	600	-	600
Initial temp.	-	-	–	-	1000	1000
Final temp.	-	-	–	-	0.5	0.5
Optimalrate	-	-	0	0	-	-

enrolment-based course timetabling problem. Details of the datasets are shown in Table 1). In this paper, all parameters used are based on a number of preliminary experiments. After the completion of the time quantum of a current algorithm, the preemption is given to the next algorithm to wait in a queue, and the next algorithm will start with best solution (*Solbest*). The pre-empted algorithm is then placed at the back of the queue, and its parameters are reset. The parameters involved in the great deluge algorithm are the estimated quality (coded as *Optimalrate*) and time to spend (coded as *NumOfIteGD*) as in Figure 2. When the algorithm ALGi is unable to generate a better solution during the given quantum time, the algorithm will be added into the queue. In the next iteration (Figure 1), the first algorithm in the queue will be used to generate a new solution. Table 1 shows the parameter setting (which is based on preliminary experiments) used in this work for different sets of data tested here.

Hill climbing algorithm

In the hill climbing algorithm, the generated initial solution *Sol* is assigned as a current solution (denoted as Sol_{Hill}) and best solution (denoted as $Sol_{bestHill}$). In each of the iteration, two neighborhood structures are employed on Sol_{Hill} to generate two new solutions

```
Initialization
  SolGD ← Sol;
  SolbestGD ← Sol;
  f(SolGD) ← f(Sol);
  f(SolbestGD)← f(Sol)
  Set optimal rate of final solution, Optimalrate;
  Set number of iterations, NumOfIteGD;
  Set initial level: Level ← f(SolGD);
  Set decreasing rate ΔB= ((f(SolGD)–Optimalrate)/(NumOfIteGD);
  Set iteration ← 0;
  Set not_improving_counter ← 0, not_improving_length_GDA;
Improvement
  Do while (not termination criteria)
    Apply neighbourhood structure $N_i$ where $i \in \{1,2\}$ on SolGD,TempSolGD$_i$;
    Calculate cost function f(TempSolGD$_i$);
    Find the best solution among TempSolGD$_i$ where $i \in \{1,2\}$ call
    new solution SolGD*;
    if (f(SolGD*) < f(SolbestGD))
      SolGD ← SolGD*;
      SolbestGD ← SolGD*;
      not_improving_counter ← 0;
      level = level - ΔB;
    else
      if (f(SolGD*)≤ level)
        SolGD ← SolGD*;
        not_improving_counter ← 0;
      else
        not_improving_counter++;
        if (not_improving_counter == not_improving_length_GDA)
```

Figure 2. The pseudo code for the Great Deluge.

```
SolSA ← Sol;
SolbestSA ← Sol
f(SolSA) ← f(Sol);
f(SolbestSA)← f(Sol)
Set initial temperature $T_0$
Set final temperature $T_f$;
Set decreasing rate $\alpha$ = (log ($T_0$) - log ($T_f$)/Iter_max);
Set not_improving_counter ← 0
Set not_improving_ length_SA;

do while (not termination criteria)
    Define a neighbourhood $N_i$ where $i \in \{1, 2\}$ on SolSA to generate TempSolSA$_i$;
    Calculate cost function f(TempSolSA$_i$);
    Find the best solution among TempSolSA$_i$ where $i \in \{1, 2\}$
    call new solution SolSA*;
      if (f(SolSA*) < f(SolbestSA))
        SolSA ← SolSA*;
        SolbestSA ← SolSA*;
      else
        not_improving_counter++;
        if (not_improving_counter == not_improving_length_SA)
          Generate a random number, RandNum in [0, 1];
          Calculate the acceptance propability of SolSA*, Paccept(SolSA*)
          if (RandNum < $P_{accept}$(SolSA*)) // $P_{accept}$(SolSA*) is a function to calculate the acceptance probability of SolSA*
             SolSA ← SolSA*;
          End if
      end if
      temp ← temp/(1+ α);
  end do;
  return SolbestSA;
```

Figure 3. The pseudo code for the simulated annealing.

and the best is selected among of them, called *TempSol$_{Hill}$*. The *f*(*TempSol$_{Hill}$*) is compared to the *f*(*Sol$_{bestHill}$*). *TempSol$_{Hill}$* will be accepted when it does not worsen the overall solution value (that is, the sum of violated soft constraints). Then the current and best solutions within the hill climbing algorithm operations are updated (*Sol$_{bestHill}$* ← *TempSol$_{Hill}$*, *Sol$_{Hill}$* ← *TempSol$_{Hill}$*). The process is repeated until the termination criterion is met, then the best solution obtained from the hill climbing algorithm (denoted as *Sol$_{bestHill}$*) is returned.

Great Deluge algorithm

Great Deluge algorithm uses a bound *level* that is imposed on the overall value of the current solution that the algorithm is working with as the generated solution is only accepted when the value of the solution after applying the neighbourhood does not exceed the *level* (McMullan, 2007). Figure 2 shows the pseudo code for great deluge algorithm. Here *SolGD and SolbestGD* are set to be *Sol*. The *level* starts at value (*level* = *f*(*SolGD*)), where *SolGD* is the current solution of great deluge algorithm where *SolbestGD* is the overall value of the best solution so far. The *level* is decreased after each iteration by ΔB (ΔB = ((*f*(*SolGD*) – *Optimalrate*)/(*NumOfIteGD*)), where *Optimalrate* is the estimated quality of the final solution that a user requires.

Two of the neighbourhoods are applied to *SolGD* to obtain *TempSolGD$_i$*. The best solution among *TempSolGD$_i$* is identified, called, *SolGD**. The *f*(*SolGD**) is compared to the *f*(*SolbestGD*). If it is better, then the current and best solutions are updated. Otherwise, *f*(*SolGD**) will be compared against the level. If the quality of *SolGD** is less than the level, the current solution, *SolGD* will be updates as *SolGD**. Otherwise, the *level* will be increased with a certain number (which is set in between 1 and 3 in this experiment) in order to allow some flexibility in accepting worse solution. The process is repeated until the termination criterion is met.

Simulated annealing algorithm

Simulated annealing algorithm uses a temperature, *temp*. Here the current solution (*SolSA*) and the best solution (*SolbestSA*) are set as *Sol*. The same parameters as those employed in Abdullah et al. (2010) are used where the initial temperature T_0 is equal to 1000; the final temperature T_f is equal to 0.5. Two neighbourhoods outlined in this paper are applied to *SolSA* to obtain *TempSolSA$_i$* and choose the best among *TempSolSA$_i$* called, *SolSA**. A generated solution is accepted when it is not worsening the overall value of the current solution. Otherwise the worse solution is accepted with a probability as in Abdullah et al. (2010).

$$P_{accept} = exp(-\frac{f(SolSA^*) - f(SolSA)}{f(SolSA^*)temp})$$

Let *f(SolSA)* is the value of the current solution and *f(SolSA*)* is the value of the new solution after a number of non improvement (worse solution). Figure 3 shows the pseudo code for the simulated annealing.

Table 2. The parameter values for the course timetabling problem categories.

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

Table 3. Results comparison.

Data set	Our best method	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
Small1	0*	2	0*	6	0*	0*	0*	3	0*	0*	0*
Small2	0*	4	0*	7	0*	0*	0*	4	0*	0*	0*
Small3	0*	2	0*	3	0*	0*	0*	6	0*	0*	0*
Small4	0*	0*	0*	3	0*	0*	0*	6	0*	0*	0*
Small5	0*	4	0*	4	0*	0	0*	0*	0*	0*	0*
Medium1	117	226	242	372	317	221	80*	140	96	93	124
Medium2	108	215	161	419	313	147	105	130	96	98*	117
Medium3	135*	231	265	359	357	246	139	189	135	149	190
Medium4	75*	200	181	348	247	165	88	112	79	103	132
Medium5	160	195	151	171	292	130	88	141	87	98	73*
Large	589	1012	-	1068	-	529	730	876	683	680	424*

*The best results.

At every iteration, *temp* is decreased by α, where α is defined as (log (T_0) – log (T_f)/*Iter_max*) where Iter_max is the maximum number of iterations. The process is repeated until the termination criterion is met. The pseudo code for the simulated annealing algorithm shows in Figure 3. Based on our preliminary experiments, the counter of not improving solutions (*not_improving_length_SA*) is set to 20.

Neighbourhood moves

Two neighbourhood moves are employed in this approach 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.

EXPERIMENTAL RESULTS

The algorithm is coded using Matlab under Windows XP and performed on the Intel 2.33 GHz computer and tested on enrolment-based benchmark datasets and on curriculum-based course timetabling problems. For each benchmark data set, the algorithm was run with 11 test-runs to obtain an average value.

Enrolment-based course timetabling problem

We evaluate our proposed approach on the instances taken from Socha et al. (2002), and http://iridia.ulb.ac.be/~msampels/tt.data/. These benchmark course timetabling problems was proposed by the meta-heuristics network that need to schedule 100 to 400 courses into 45 timeslots that corresponds to 5 days of 9 h 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 2.

The best results for 200000 iterations and out of 11 runs obtained are presented. Table 3 shows the comparison of the approach in this paper with other available approaches in the literature. These include

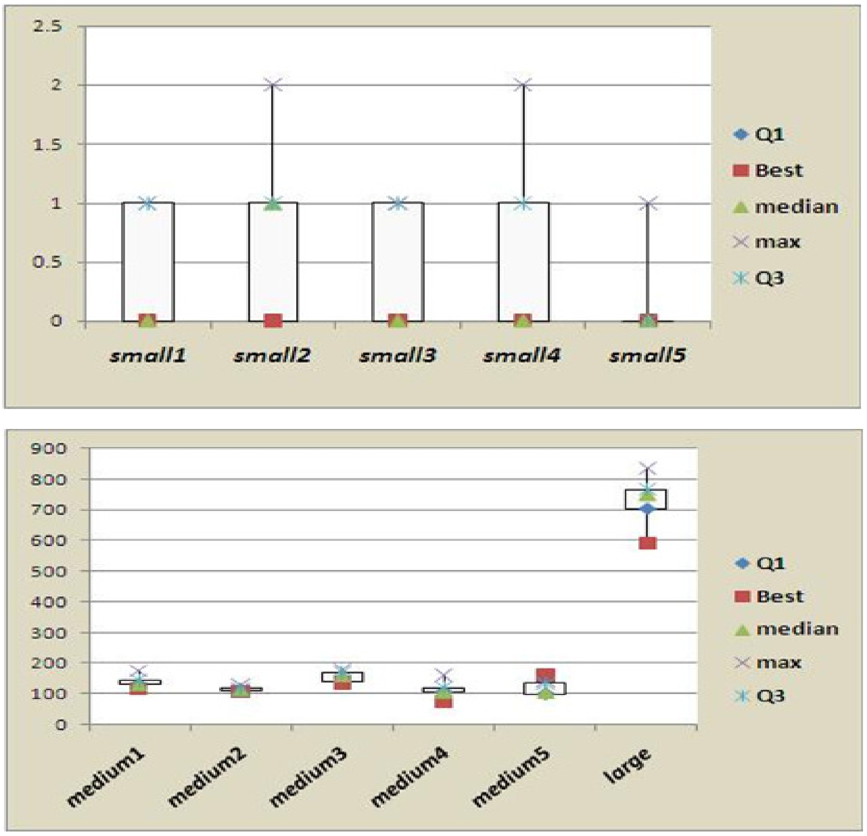

Figure 4. Box plots of the penalty costs for *small*, *medium* and *large* datasets.

genetic algorithm and local search by M1-Abdullah and Turabieh (2008), randomised iterative improvement algorithm by M2-Abdullah et al. (2007), graph hyper heuristic by M3-Burke et al. (2007), variable neighbourhood search with tabu by M4-Abdullah et al. (2005), hybrid evolutionary approach by M5-Abdullah et al. (2007), extended great deluge by M6-McMullan (2007), non linear great deluge by M7-Landa-Silva and Obit (2008), electromagnetism-like mechanism approach by M8-Turabieh et al. (2009), Dual simulated annealing by M9-Abdullah et al. (2010), and M10-Harmony search by Al-Betar et al.(2010). It can be seen that our approach is able to produce competitive results. From Table 3 it can be seen our approach has competitive results and better results on *mediums 3* and *4*.

Figure 4 shows the box plots of the penalty cost when solving small, medium and large instances. The results for the large dataset are less dispersed compared to medium and small (where *small* instance shows a worse dispersed case in these experiments).

Curriculum-based course timetabling problem

Table 4 shows the main features of these instances, including: courses (*C*), total lectures (*L*), rooms (*R*), periods permday (*PpD*), days (*D*), curricula (*Cu*), min and max lectures per day per curriculum (*MML*).

Here, same neighbourhood moves are applied on all datasets presented in Table 4. Table 5 shows the comparison between the best results obtained by our algorithm with the best known results from the literature. It can be clearly seen that the best results obtained by our approach are competitive to the previously best known results and able to obtain best result on the *comp*05, *comp*21, *DDS*2 , *DDS*3 , *DDS*4 , *DDS*5 , *DDS*6,and *Test*1-*Test4* datasets.

Figure 5 shows the convergence of the penalty cost for *comp*21 and *Test*1 datasets at every iteration. The x-axis represents the number of iterations, while y-axis represents the penalty cost. The trend of both graphs are similar which indicate that the algorithm behaves similar even though the features of the datasets maybe different.

It can be seen from the figures that the penalty cost can be quickly reduced at the beginning of the search (shown as a steep slope) which increases the diversity and gives a greater chance to find better solutions. Later, the graph looks stagnant as the evolution continue. It means that the searching process is nearly converged whilst the possibility of finding improved solutions becomes smaller.

Table 4. Description of the instances.

Instance	*C*	*L*	*R*	*PpD*	*D*	*Cu*	*MML*
Comp01	30	160	6	6	5	14	2-5
Comp02	82	283	16	5	5	70	2-4
Comp03	72	251	16	5	5	68	2-4
Comp04	79	286	18	5	5	57	2-4
Comp05	54	152	9	6	6	139	2-4
Comp06	108	361	18	5	5	70	2-4
Comp07	131	434	20	5	5	77	2-4
Comp08	86	324	18	5	5	61	2-4
Comp09	76	279	18	5	5	75	2-4
Comp10	115	370	18	5	5	67	2-4
Comp11	30	162	5	9	5	13	2-6
Comp12	88	218	11	6	6	150	2-4
Comp13	82	308	19	5	5	66	2-3
Comp14	85	275	17	5	5	60	2-4
Comp15	72	251	16	5	5	68	2-4
Comp16	108	366	20	5	5	71	2-4
Comp17	99	339	17	5	5	70	2-4
Comp18	47	138	9	6	6	52	2-3
Comp19	74	277	16	5	5	66	2-4
Comp20	121	390	19	5	5	78	2-4
Comp21	94	327	18	5	5	78	2-4
Dds1	210	900	21	15	5	99	3-7
Dds2	82	146	11	11	6	11	3-6
Dds3	50	206	8	11	5	9	3-6
Dds4	217	972	31	10	5	105	3-6
Dds5	109	560	18	12	6	44	3-6
Dds6	107	324	17	5	5	62	2-4
Dds7	49	254	9	10	6	37	3-6

Table 5. Best results and comparison with other algorithms.

Dataset	Our method	M1	M2	M3	M4	M5	M6	M7	M8
Comp01	5*	5*	13	5*	5*	9	5*	5*	5*
Comp02	43*	43*	43	75	34	103	108	50	60
Comp03	77	72	76	93	70*	101	115	82	81
Comp04	38	35*	38	45	38	55	67	35	39
Comp05	311*	298	314	326	298	370	408	312	321
Comp06	44	41*	41	62	47	112	94	69	45
Comp07	19	14*	19	38	19	97	56	42	21
Comp08	44	39*	43	50	43	72	75	40	41
Comp09	108	103*	102	119	99	132	153	110	102
Comp10	13	16	14	27	16	74	66	9	17
Comp11	0	0	0	0	0	1	0*	0*	0*
Comp12	339	331	405	358	320*	393	430	351	349
Comp13	69	66	68	77	65	97	101	68	73
Comp14	60	53	54	59	52*	87	88	59	59
Comp15	76	84	-	87	69*	119	128	82	82
Comp16	48	34	-	47	38	84	81	40	49

Table 5. Contd.

Comp17	91	83	-	86	80*	152	124	102	81
Comp18	84	83	-	71	67*	110	116	68	79
Comp19	71	62	-	74	59	111	107	75	67
Comp20	42	27	-	54	35	144	88	61	30
Comp21	103*	103	-	117	105	169	174	123	110
Dds1	143	-	132*	1024	-	-	-	-	158
Dds2	0	-	0*	0	-	-	-	-	0*
Dds3	0	-	0*	0	-	-	-	-	0*
Dds4	24	-	68	233	-	-	-	-	28
Dds5	0	-	0*	0	-	-	-	-	0*
Dds6	4	-	4*	11	-	-	-	-	4*
Dds7	0	-	0*	0	-	-	-	-	0*
Test1	227*	-	-	234	-	-	-	-	-
Test2	16*	-	-	17	-	-	-	-	-
Test3	83*	-	-	86	-	-	-	-	-
Test4	89*	-	-	132	-	-	-	-	-

M1: A constraint-based solver by Müller (2009); M2: Integer programming by Lach and Lübbecke (2010); M3: The dynamic tabu search by Cesco et al. (2008); M4: Adaptive tabu search by Lü and Hao (2010); M5: A repair-based timetable solver by Clark et al. (2008); M6: Threshold accepting met-heuristic by Geiger (2010); M7: Incorporating tabu search and iterated local search by Atsuta et al. (2008); M8: Great Deluge approach with Kempe Chain by Shaker and Abdullah (2009); *The best results.

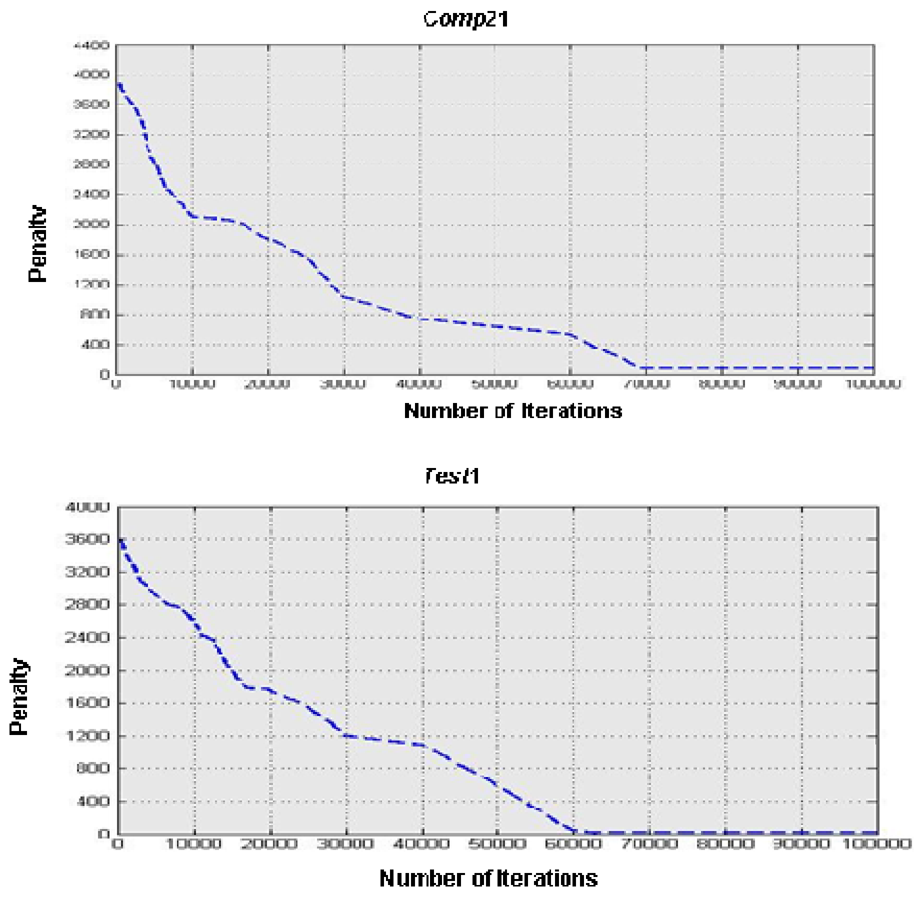

Figure 5. Convergences of *comp*21 and *test* 1 datasets.

CONCLUSION AND FUTURE WORK

This paper presents hill climbing, great deluge and simulated annealing algorithms applied to the course timetabling problem. The round robin algorithm is employed on these algorithms to control the selection of the algorithms given a slice time or quantum. In order to test the performance of our approach, experiments are carried out based on course timetabling problems and compared with state-of-the-art methods from the literature. Preliminary comparisons indicate that our approach is competitive with other approaches in the literature and able to produce two best known solutions on mediums 3 and 4 dataset and best known result on the *comp*05, *comp*21, *DDS*2, *DDS*3, *DDS*4, *DDS*5, *DDS*6, and *Test*1-*Test4* datasets. In future work, efforts will be made to establish and compare in relation to previously reported literature. We believe that the proposed approach can be adapted with new problems, thus the Track 2 of ICT2007 will be the subject of future work.

REFERENCES

Abdullah S, Turabieh H (2008). Generating university course timetable using genetic algorithms and local search. The Third int. Conf. Convergence Hybrid Inf. Technol., ICCIT, 1: 254-260.

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

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

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

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

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

Asmuni H, Burke EK, Garibaldi JM (2005). Fuzzy multiple heuristic ordering for course timetabling. The Proceedings of the 5th United Kingdom Workshop on Computational Intelligence (UKCI05), pp. 302-309.

Burke EK, Bykov Y, Newall J, Petrovic S (2003). A Time-Predefined Approach to Course Timetabling. Yugoslav J. Oper. Res., (YUJOR), 13(2): 139-151.

Burke EK, Kendall G, Soubeiga E (2003). A tabu search hyperheuristic for timetabling and rostering. J. Heuristics, 9(6): 451-470.

Burke EK, Mareˇcek J, Parkes AJ, Rudová H (2009). Decomposition, reformulation, and diving in university course timetabling, Comp. Oper. Res., doi:10.1016/j.cor.2009.02.023.

Burke EK, McCollum B, Meisels A, Petrovic S, Qu R (2007). A Graph-Based Hyper Heuristic for Educational Timetabling Problems, Eur. J. Oper. Res., 176(1): 177-192.

Cesco De F, Di Gaspero L, Schaerf A (2008). Benchmarking curriculum-based course timetabling: Formulations, data format, instances, validation and results. In Proceedings of the 7th PATAT Conference.

Chiarandini M, Birattari M, Socha K, Rossi-Doria O (2006). An effective hybrid algorithm for university course timetabling. J. Scheduling, 9(5): 403-432

Clark M, Henz M, Love B (2008). QuikFix: A repair-based timetable solver. In Proceedings of the 7th PATAT Conference.

Di Gaspero L, McCollum B, Schaerf A (2007). The Second International Timetabling Competition (ITC2007): Curriculum-based Course Timetabling Track3, the 14th RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion.

Dimopoulou M, Miliotis P (2004). An automated university course timetabling system developed in a distributed environment: A case study. Eur. J. Oper. Res., 153(1): 136-147.

Gaspero LD, Schaerf A (2003). Multi-neighbourhood local search with application to course timetabling. In Emund Burke and Patrick De Causmaecker, editors. Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT-2002), selected papers, volume 2740 of Lecture Notes in Computer Science, Springer, pp. 262–275.

Geiger MJ (2008). An application of the threshold accepting metaheuristic for curriculum-based course timetabling. In Proceedings of the 7th PATAT Conference.

Glover F, Laguna M (1997). Tabu Search. Kluwer Academic, Boston

Kostuch P, Socha K (2004). Hardness Prediction for the University Course Timetabling Problem. Proceedings of the Evolutionary Computation in Combinatorial Optimization (EvoCOP 2004), Coimbra, Portugal. Springer Lecture Notes Comput. Sci., 3004: 135-144.

Kostuch P (2005). The university course timetabling problem with a three-phase approach. Practice and Theory of Automated Timetabling V (eds. Burke and Trick), Springer Lecture Notes in Comput. Sci., 3616: 109-125.

Lach G, Lubbecke ME (2008). Curriclum-based course timetabling: Optimal solutions to the udine benchmark instances. In Proceedings of the 7th PATAT Conference.

Landa-Silva D, Obit JH (2008). 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.

Lewis R, Paechter B (2004). 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.

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

Lu Z, Hao J (2009). Adaptive Tabu Search for Course Timetabling. Eur. J. Oper. Res., doi:10.1016.j.ejor.2008.12.007.

Lu Z, Hao J (2008). Solving the Course Timetabling Problem with a Hybrid Heuristic Algorithm. AIMSA 2008, LNAI, Springer-Verlag Berlin Heidelberg, 5253: 262–273.

McCollum B (2007). A perspective on bridging the gap between theory and practice in university timetabling. LNCS 3867, Springer-Verlag, 3-23.

McCollum B, Burke EK, McMullan P (2010). A review and description of datasets, formulations and solutions to the University Course Timetabling Problem. To be submitted April 2010 to the J. Scheduling.

McMullan P (2007). An Extended Implementation of the Great Deluge Algorithm for Course Timetabling. Lecture Notes Comput. Sci., Springer, 4487: 38-545.

Müller T (2008). ITC2007: Solver Description. Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling.

Ross P, Corne D (1995). Comparing genetic algorithms, simulated annealing, and stochastic hill climbing on timetabling problems. In: GOOS, G. HARTMANIS, J. and LEEUWEN, J. (eds.) Evolutionary Computation, AISB Workshop, 94–102. Lecture Notes in Computer Science, Springer-Verlag, Sheffield, p. 993.

Shaker K, Abdullah S (2009). Incorporating Great Deluge Approach with Kempe Chain Neighbourhood Structure for Curriculum-Based Course Timetabling Problems. 2nd IEEE Conference on Data Mining and Optimization (DMO'09), pp. 149-153.

Socha K, Knowles J, Samples M (2002). 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 Comput. Sci., 2463: 1-13.

Turabieh H, Abdullah S (2009). McCollum, B.: Electromagnetism-like Mechanism with Force Decay Rate Great Deluge for the Course Timetabling Problem. In: RSKT 2009. LNCS, Springer, Heidelberg, 5589: 497–504.