International Journal of Computer Applications (0975 - 8887)
Volume 124 - No.10, August 2015

Genetic Algorithm for Solving Course Timetable Problems

Mahmoud M. El-Sherbiny
Operations Research Dept., Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt

Ramadan A. Zeineldin
Operations Research Dept., Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt

Abdallah M. El-Dhshan
Operations Research Dept., Institute of Statistical Studies and Research (ISSR), Cairo University, Egypt

ABSTRACT

Course Timetable Problem (CTTP) is considered as a multi-assignments problem. This problem occurs during the assignment of a set of events (courses, subjects and teachers) to specific number of appointments (timeslots, days and rooms) under a set of hard and soft constraints. In this paper, the proposed algorithm is represented for solving the CTTP based on the combination of Genetic Algorithms (GA) and Hill Climbing Optimization (HCO). The proposed algorithm is tested over two stages. The first stage is used to discover the fittest mutation out of 16 mutation functions. So, the Relative Percentage Deviation (RPD) is described as a comparison method to discover the best mutation function for solving the CTTP. The second stage is considered to measure the effectiveness and efficiency of the proposed algorithm over 5 datasets namely hddt benchmark. The results show that the proposed algorithm is able to generate good and optimal solutions when compared against other approaches from literature.

General Terms

Genetic Algorithms, Hill Climbing Optimization

Keywords

Course Timetable Problem, Genetic Algorithms, Hill Climbing Optimization, Metaheuristics

1. INTRODUCTION

Timetabling is an important daily life problem that is frequently encountered in educations, companies, sports event and transport. Course Timetable Problem (CTTP) is a major type of education filed, The CTTP classified as NP-complete problem as it incorporation multi-dimensional and multi-objectives. Generate a solution for timetabling by traditional methods need a very difficult effort and time to satisfy all problem constraints. The problem constraints can be classified as hard constraints which must not be violated because it effect on the solution feasibility, and soft constraints which can be violated if necessary, but will effect on the solution quality. The objective of timetabling problem is to maximize the number of stratified soft constraint for a feasible solution that is generated by the hard constraints. As the CTTP is a one of the reachable development area, there are many researches cared it out, and varies of methodologies are used for solve this type of problems. Metaheuristics is a fertile field for optimizing the CTTP. The metaheuristics can be divided into Singlebased (Search Trajectory) and Population-based algorithms [11]. There are many single-based algorithms implemented for solving the CTTP such as tabu search [12], simulated annealing [20] and great deluge [18]. Also, the population-based algorithms are used for solving the CTTP for example genetic algorithm [13][15][16] and particle swarm [10][6]. In addition, other techniques are presented such as heuristic [15], hybridized approaches [4][3] and integer programming as a one of methodologies that are used to solve CTTP [14].

The main contribution of the study presented in this work to propose an algorithm able to solve the course timetable problems and Compete with the other techniques. The proposed algorithm combines the Genetic Algorithm (GA) and Hill Climbing Optimization (HCO).The GA as a type of metaheuristic approaches explores the solutions of the search space which guided to give optimal or near optimal solutions. While, the HCO is used as a local search method to move form a solution to a better one. In addition to GA and HCO, A powerful mutation type selected from 16 types of mutations applied in the reproduction phase. Also, A new coding and decoding functions used in the solution representation. This proposed algorithm was developed to produce better results than other methods solved the hdtt benchmark which contains a five problems.

The paper is organized as follows. Section 1 is the introduction. In Section 2, the problem definition of CTTP is described. In Section 3, it is focused on the solution representation. In Section 4, the proposed algorithm combining genetic algorithm and hill climbing to solve the CTTP is described. In Section 5, shows the testing and the experimental results of the proposed algorithm. Finally, conclusions are included in Section 6.

2. PROBLEM DEFINITION

As described in the introduction, CTTP is known as a multi-dimensional assignment problem in which teacher assigned to class to teach subject in a room at a specific time. The assignment has to satisfy a set of hard and soft constraints.

Let T is a set of teachers; each lecture of each subject s will have a teacher previously assigned.

let C a set of classes; each class has a fixed number of lectures during a week; and the class c must assigned to only one teacher for any timeslot.

Let S a set of subjects.

Let R a set of rooms.

Let P a set of teaching periods per day.
Let D a set of days.
Let $A_{t,s}$ a set of periods when teacher t of subject s is available.
And let the decision variables be as:

$$X_{tcsrpd} = \begin{cases} 1 & \text{if teacher } t \text{ teaches subject } s \text{ for class } c \text{ in room } r \\ & \text{at period } p \text{ in day } d \\ 0 & \text{otherwise} \end{cases}$$

2.1 Hard Constraints

The hard constraints are the pivot element of building the CTTP. Any feasible solution has to satisfy all hard constraints. The set of CTTP hard constraints follows as [20]:
HC1:A teacher cannot be assigned to more than one class during any time slot.

$$\sum_{c \in C} X_{tcsrpd} \leq 1 \quad \forall\, t \in T_s, s \in S, r \in R, p \in P, d \in D, \quad (1)$$

HC2: A class cannot be assigned to more than one teacher for any time slot.

$$\sum_{s \in S} \sum_{t \in T_s} X_{tcsrpd} \leq 1 \quad \forall\, c \in C, r \in R, p \in P, d \in D, \quad (2)$$

HC3: A room cannot be allocated to more than once for any time slot.

$$\sum_{t \in T_s} \sum_{c \in C} \sum_{s \in S} X_{tcsrpd} \leq 1 \quad \forall\, r \in R, p \in P, d \in D, \quad (3)$$

HC4: A teacher t, teaching subject s, is to be assigned $l_{t,s}$ lectures (time periods or time slots) per week.

$$\sum_{c \in C} \sum_{r \in R} \sum_{d \in D} \sum_{p \in P} X_{tcsrpd} = l_{t,s} \quad \forall\, s \in S, t \in T_s, \quad (4)$$

HC5: A class c must attend l_c lectures per week.

$$\sum_{s \in S} \sum_{t \in T_s} \sum_{r \in R} \sum_{d \in D} \sum_{p \in P} X_{tcsrpd} = l_c \quad \forall\, c \in C, \quad (5)$$

2.2 Soft Constraints

The soft constraints are used as an evaluation tool quality of a feasible timetable. They are different from problem to another upon the enterprise rules such as teacher preference time, consecutive and isolated lectures for a students, and room utilization.
SC1: Teachers' timeslot preferences is considered to assign a set of Teachers to a set of timeslots based on the Teachers' preferences. Every teacher lists the timeslot preferences to teach class. Let $T = \{1, 2, \cdots, t\}$ be a set of teachers, and $P = \{1, 2, \cdots, p\}$ be a set of timeslots, where (t≤p). c_{ij} is defined as the preference given by teacher i to being assigned j timeslot. A teacher-timeslot preference matrix $(t \times p)$ is formed to contain the teachers preferences in the following sense. The value of $'1'$ indicates the first timeslot choice of the teacher, the value of $'2'$ indicates the second timeslot choice, and so on up to a specific number indicating the maximum number of timeslots $'p'$. If c_{ij} has not been assigned an integer value (i.e. teacher i has not included timeslot j in their list of preferences), then c_{ij} is assigned a penalty value B (suitably large). Additionally, some priority weights w_i could be assigned to each teacher so as to give some teachers a better chance of being allocated their higher preference projects. The mathematical programming formulation can be illustrated as follows [8]:

$$Min \sum_{i=1}^{t} \sum_{j=1}^{p} w_i c_{ij} x_{ij} \quad (6)$$

$$s.t. \quad \sum_{j=1}^{t} x_{ij} = 1 \quad \forall\, i \in p \quad (7)$$

Equation (7) ensures that each teacher is assigned only one timeslot.

$$\sum_{i=1}^{t} x_{ij} \leq 1 \quad \forall\, j \in p \quad (8)$$

Equation (8) specifies that each timeslot is assigned at most once as the number of timeslots might exceed the number of teachers.

$$x_{ij} \in \{0, 1\} \quad \forall\, i \in p, \quad \forall\, j \in T \quad (9)$$

Where

$$X_{ij} = \begin{cases} 1 & \text{if teacher } i \text{ is assigned to timeslot j} \\ 0 & \text{otherwise} \end{cases}$$

and
$\forall\, i \in P,\ j \in T \quad ,\ c_{ij} = B \quad if\ \ c_{ij} \notin T$

SC2: Room Utilization
Let $R = \{1, 2, \cdots, m\}$ be a set of rooms, $D = \{1, 2 \cdots, y\}$ be a set of days and $P = \{1, 2, \cdots, n\}$ is a set of teaching timeslots for ever day. The capacity of rooms is RC_r and let SN_{rdt} be number of students attending a course assigned to room r during timeslot p at day d. rooms' utilization can be formulated as minimization a cost function. The cost function is calculated by the difference between the number of students and room capacity. The mathematical formulation as follows [2]:

$$Min \sum_{r=1}^{m} \sum_{d=1}^{y} \sum_{p=1}^{n} (RC_r - SN_{rdp}) \quad (10)$$

$$s.t. \quad \sum_{r=1}^{m} SN_r \leq Ri \quad and \quad SN_{rdp}\ \ exists \quad (11)$$

2.3 Fitness Function

Timetable schedule should have a fitness value as evaluation of an individual's fitness which is labeled fitness function. It can be represented as [7]:

$$eval(f) = \frac{1}{1 + cost(f)} \quad (12)$$

Where cost(f) is value of violating constraints and is calculated as (13):

$$cost(f) = \sum_{i=1}^{ct} n_i(f) \times W_i \quad (13)$$

Where ct is the count of constraints, $n_i(f)$ is the penalty of violating constraint i and w_i is its weight.

The objective of CTTP is to minimize the number of soft constraints violation in a feasible solution that generated by hard constraints.

3. SOLUTION REPRESENTATION

A good structure of the solution is important in GA. It makes the generation and reproduction of the solutions more easier, and helpful in solving CTTP. Fig.1 illustrates how timetable schedule (individual) is represented. Appointment $A = \{a_1, a_2, \cdots, a_j\}$ represented as set of periods (timeslots) $P = \{1, 2, \cdots, p\}$, for every room $R = \{1, 2, \cdots, r\}$, every day $D = \{1, 2, \cdots, d\}$. Appointments can be represented as a vector with size $(p \times r \times d)$. There is an additional list of events $E = \{e_1, e_2, \cdots, e_i\}$ which is used to link subject $S = \{1, 2, \cdots, s\}$, teacher $T = \{1, 2, \cdots, t\}$, and class (student group) $C = \{1, 2, \cdots, c\}$.

Events	e_1	e_2	e_3			e_{i-1}	e_i
Appointments	3	4	19	a_j	...	16	77

Fig. 1. Representation of a chromosome.

The available appointment list is an array of length $[(roomNum \times dayNum \times periodNum)]$ its index designed to know the details of the appointment (which room, day and period) by using the index encoding and decoding. Index design makes the reproduction of the chromosome more easier.

3.1 Index Encoding

The index can be calculated by (14).

$$Index = (r \times Days\# \times Periods\#) + (d \times Periods\#) + p \quad (14)$$

Where r**:** room number, d**:** day number, p**:** period number, $Days\#$**:** Number of days per week, $Period\#$**:** Number of periods per day.
Example: If $Rooms\#$ = 3 rooms, $Days\#$ = 5 days, and $Periods\#$ = 12 period per day. So, the length of the appointment array will be:
length=$[3 \times 5 \times 12]$ (**180** appointments)
If r = 2, d =2 and p =9. Then the allocation index will be:
Index=$(2 \times 5 \times 12) + (2 \times 12) + 9 =$ **153**

3.2 Index Decoding

Reverse calculation of index to get timeslot, room and day can be as (15), (16) and (17) respectively:

$$p = index \mod Periods\# \quad (15)$$

$$d = (index/Periods\#) \mod Days\# \quad (16)$$

$$r = (index/(Periods\# \times Days\#)) \mod Rooms\# \quad (17)$$

Where, $\mod$ is modulo operation.
Example: If the index =162 then we can calculate period, room and day as follows:
$p = 162 \mod 12 = 6$,
$d = (162/12) \mod 5 = 3$,
$r = (162/(12 \times 5)) \mod 5 = 2$.

4. THE PROPOSED ALGORITHM

In this section, the proposed algorithm is adding a short memory wise with hill climbing to GA platform to find optimal solutions for CTTP. The proposed algorithm developed to reduce the time of GA with the powerful of hill climbing as a local search technique. In addition, the Hill Climbing Rate (HCR) is used to control the Hill Climbing Optimization (HCO) to avoid trapping into local optimal solution.

The proposed algorithm

Step 1: Identify the number of generation and the population size.
Step 2: Declare an initial individual to be the best individual.
Step 3: Set g =1.
Step 4: Create initial population of individuals.
Step 5: For each individual in the population.
  5.1. Evaluate the individual.
  5.2. If the fitness of the individual is better than the best individual.
  5.3. Replace the best individual by the current individual.
  5.4. If the best individual is idle solution go to step 10.
Step 6: For population size times do.
  6.1. Select individual from the population.
  6.2. Apply the mutation function on the selected individual using hill climbing Optimization.
  6.3. Add the mutated individual to the new population.
Step 7: Update the population with the new population.
Step 8: g =g+1.
Step 9: Repeat step 5 to step 8 until g > the number of generation.
Step 10: Return the best individual.

4.1 Initial Population

As described in solution representation, there are group of events (Teacher Id, Class Id and Subject Id) need an appointments (Room Id, Day Id and Period Id) to create the chromosome (individual or solution). For each new individual the proposed algorithm orders the events by its number of lectures, then start with the first event and select randomly an appointment from the free appointments list (dynamic list for only the free appointments) and allocate the index of the appointment to the event in the individual. In Parallel, the proposed algorithm uses a short memory wise to tabu the appointments for each teacher and class to prevent the clashing as can possible by comparing this list with the free appointments list at the next event to get a fit assignment.

4.2 Evaluating Fitness Function

Fitness function checks the constraints violation for every event in the individual, and marks this event as a violation event that will used in the reproduction population in the next section, the number of violating event called the cost function and use fitness function equation in section 3 to calculated the fitness of the individual.

4.3 Reproduction

As the solution structure that is described in section 3, the crossover operation will be difficult to use and will not produce a good children than their parents. So, the proposed algorithm depends on the mutation operation only. Therefore, 16 type of mutation functions are represented as following to determined the fittest one for solving CTTP [9].

(1) Two point swaps which based on select a random conflicted event and swap with another random event. Then, swap the appointment of each other Fig.2

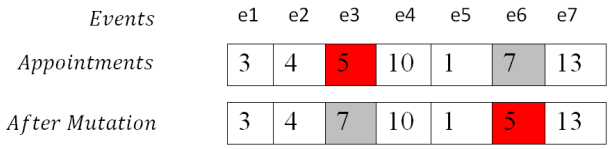

Fig. 2. Two point swap.

(2) Inverse swap, this type of mutation is based on select two random events, one from the conflicted events and the other from all events. Then, inverse the order of the appointments between the two events Fig.3

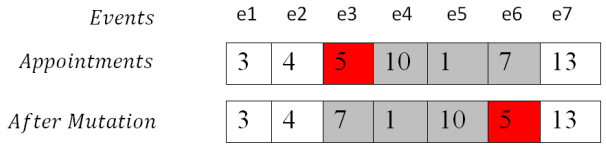

Fig. 3. Inverse swap.

(3) Neighbor swap, which based on select a random conflicted event and swap appointment with the next event Fig. 4.

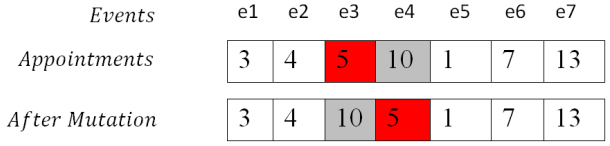

Fig. 4. Neighbor swap.

(4) Uniform random number swap, which swaps with a setup fixed number for all individuals during all iterations.This fixed number is donated by MNS.The number of swaps (NS) for this mutation is represented by (18).

$$NS = MNS \tag{18}$$

(5) This mutation is based on generating a random number for $NS \in [1, a]$ where a denotes to the length of the available events.Then, the two point swap (mutation number 1) is performed NS times. The number of swaps (NS) for this mutation is represented in (19)

$$NS = Int(Rand(1, a)) \tag{19}$$

(6) This type of mutation based on a uniform random number located in the range of 10-30% of the number of events for the individual. The number of swaps (NS) for this mutation is represented by (20), where r is a random number in the interval [0.1, 0.3].

$$NS = Int(Rand(1, r(a))) \tag{20}$$

(7) This mutation is based on time where there need more number of swaps at the beginning. And it decrease by the time elapsed. First start with applying random number of two-points-swap till a predefined ratio of time is elapsed. After that the two points swap (mutation number 1) is applied for the remaining time. The time is represented by the ratio of current iteration to the total number of iterations.

(8) This mutation is based on either non-uniform swap times or inverse swap (Mutation number 3). A random number $r \in [0, 1]$ is generated, if $r >$ the predefined value, then the non-uniform swap time will be applied; else inverse swap will be applied.

(9) Also this mutation depends on the time. A random number of swaps are applied first; then inverse swap is applied for the reminding time.

(10) This mutation is applied the neighbor swap (Mutation number 3) r times, where $r \in [0, a/2]$ for each individual.

(11) This mutation is also based on time. First applied a random number of neighbor swap till a predefined ratio of time elapsed, then the master neighbor swap (mutation number 3) is applied for the remaining time.

(12) This mutation is based on the fitness of the solution. That the Normalized Fitness (NF) of the solution increase of decrease the number of swaps abound the NF. That is for minimize problems if the NF is near to one then it need more swaps otherwise the NS will decrease the number of swaps. The NS is adopted in (21) and NF is calculated as (22).

$$NS = MNS^{(1-(1-NF)^u)} \tag{21}$$

$$NF = \frac{LowestFitness - Fitness}{LowestFitness - HighestFitness} \tag{22}$$

(13) This Mutation is designed to applying a number of swaps depends on the ratio (T) of the current iteration number (CIN) and the Total Number of Iteration (TNI), the Number of Swaps (NS) is represented in (23) where u is the degree of non-uniformity.

$$NS = MNS^{(1-T^u)}, \quad where\, T = \frac{CIN}{TNI} \tag{23}$$

(14) This mutation is based on both time and normalized fitness of the individual. It mainly depends on the average of them, it can calculate as in (24) and NS is represented in (25).

$$TF = \frac{1}{2}(NF + (1 - T)) \tag{24}$$

$$NS = MNS^{(1-TF^u)} \tag{25}$$

(15) This type of mutation is based on the non-uniform factor and the fitness but with a random factor R ∈ [0,1], which add some randomization.The mutation is adapted as (26).

$$NS = MNS \times R^{(1-NF)^u} \tag{26}$$

(16) This mutation is the same like mutation number 15 except it depends on the time not on fitness. NS is represented as (27).

$$NS = MNS \times R^{(T^u)} \tag{27}$$

4.4 Hill Climbing Optimization (HCO)

HCO is an optimization local searcher. The HCO accept only the solution that is fitter than the best one. the proposed algorithm is used it to find the local optima from the selected neighborhoods. In addition to HCO the HCR is used to exploit the neighborhoods, which controls the HCO by comparing a uniform random number with HCR parameter. That decide whether the HCO will apply on the selection neighborhood or not.

Table 1. **The comparative results of the average RPD for the proposed mutation functions**

Mutation Function	Average RPD of the instances					Overall Mean RPD	Ranks
	HDTT4	HDTT5	HDTT6	HDTT7	HDTT8		
1	0.00	0.00	0.19	0.45	0.45	0.22	10
2	0.16	0.42	0.52	0.58	0.59	0.45	12
3	19.69	13.66	9.84	6.86	5.15	11.04	15
4	0.00	0.06	0.00	0.00	0.11	0.04	5
5	**0.00**	**0.00**	**0.00**	**0.00**	**0.12**	**0.02**	**1**
6	0.00	0.00	0.00	0.14	0.29	0.09	8
7	**0.00**	**0.00**	**0.00**	**0.02**	**0.11**	**0.03**	**3**
8	0.00	0.00	0.06	0.14	0.23	0.09	9
9	0.13	0.38	0.69	0.71	0.69	0.52	13
10	20.06	13.52	9.51	6.73	5.19	11.00	14
11	20.16	13.57	9.50	6.93	5.21	11.07	16
12	0.00	0.00	0.15	0.44	0.52	0.22	11
13	0.00	0.00	0.00	0.07	0.13	0.04	6
14	0.00	0.00	0.00	0.08	0.17	0.05	7
15	0.00	0.00	0.00	0.10	0.07	0.04	4
16	**0.00**	**0.00**	**0.00**	**0.02**	**0.11**	**0.03**	**2**

5. EXPERIMENTAL RESULTS

OR Library present the HDTT-Instances which available on (http://people.brunel.ac.uk/mastjjb/jeb/orlib/tableinfo.html), the hdtt refer to the instances are hard timetabling problems as all periods must be utilized with very little or no options for each allocation. Each week is comprised of five days with six periods a day with a total of 30 timetable periods. The proposed algorithm was implemented in Visual Studio 2010 with C# language. All run times were measured using an Intel Core 2 Duo with 2.27 GHz (with the program running on one core only) with 2GB RAM. The experimental divided into two stages. The first stage is to determine the fittest mutation type of the 16 mutation types that are described in section 4 for solving the CTTP, and the second stage is to compare the proposed algorithm and the other methods.

5.1 First Experiment

While the proposed algorithm is depends only on the mutation operation to reproduce the individuals. So, it is important to discover the best mutation type of the 16 types of mutation. As the scale of is different, and they could not be compared directly. Relative Percentage Deviation (RPD) is used to compare every instance with every mutation type. RPD can be compute by using (28).

$$RPD = \frac{Alg_{sol} - Min_{sol}}{Min_{sol}} \times 100 \tag{28}$$

Where, Alg_{sol} is the solution for a specific run, and Min_{sol} is the best solution obtained out of a set of runs of a particular instance. 10 runs executed for each instance with each mutation type. The experiment parameters that are used in the runs are (10) Population size, (10000) Generation number and (0.01) hill climbing rate. RPDs and the average of the RPD of the runs of each instance for the 16 mutation functions are presented in Table1.
Based on the results presented in Table 1 and the plot in Fig.5, the mutation function number 5 is providing the minimum value as compared with the other mutation functions. It followed by mutations number 7 and 16. Since mutations number 11,3,10 and 9 are providing the largest values. So, the mutation number 5 is selected to used for solving the CTTP.

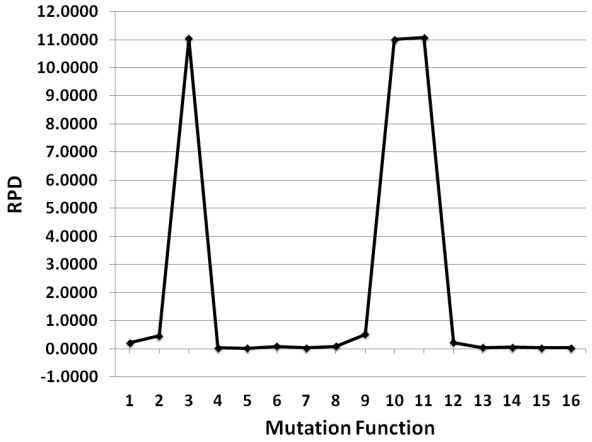

Fig. 5. Fitted mean plot for RPD at each mutation function.

5.2 Second Experiment

The proposed algorithm described in section 4 were applied to solve the five benchmark instances. The parameter values that are used for solving the CTTP are (10) population size, (20000) generation number, (0.01) hill climbing rate (HCR). Based on the results of the first experimental in section 5.1, the proposed algorithm were applied the mutation number 5.
The proposed algorithm are compared with other literature methods used to solve the CTTP for the same benchmark instances which described as follows:
SA1 Simulated annealing algorithm[1].
SA2 Simulated annealing heuristic[17].
TS Tabu Search and ***GS*** a Greedy Search are cited in [19].
NN-T2 and ***NN-T3*** A two hopfield neural network methods[19].
DWTAN and ***CPMF*** A neural network approaches described in [5].
SA3 A simulated annealing approach developed by Zhang et al.[20].
TFH Timeslot-filling heuristic and ***EAH*** event-assignment heuristic are a two heuristic methods described in [15].

Table 2. **Comparative results for the hdtt instances for the proposed algorithm and each method**

Method	HDTT4		HDTT5		HDTT6		HDTT7		HDTT8	
	Best Cost	Avg Cost	Best Cost	Avg Cost	Best Cost	Avg Cost	Best Cost	Avg Cost	Best Cost	Avg Cost
SA1	-	-	0.0	0.7	0.0	2.5	2.0	2.5	2.0	2.5
SA2	0.0	0.0	0.0	0.3	0.0	0.8	0.0	1.2	0.0	1.9
TS	0.0	0.2	0.0	2.2	3.0	5.6	4.0	10.9	13.0	17.2
GS	5.0	8.5	11.0	16.2	19.0	22.2	26.0	30.9	29	35.4
NN-TT2	0.0	0.1	0.0	0.5	0.0	0.8	0.0	1.1	0.0	1.4
NN-TT3	0.0	0.5	0.0	0.5	0.0	0.7	0.0	1.0	0.0	1.2
CPMF	5.0	10.7	8.0	13.2	11.0	18.7	18	25.6	15.0	28.6
DWTAN	0.0	0.0	0.0	0.4	0.0	1.65	0.0	2.1	0.0	3.25
SA3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.4**
EAH	0.0	0.0	2.0	5.4	6.0	7.9	9.0	12.0	13.0	15.0
TFH	0.0	0.0	0.0	0.6	0.0	2.1	0.0	2.5	0.0	3.1
Proposed Algorithm	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.6**

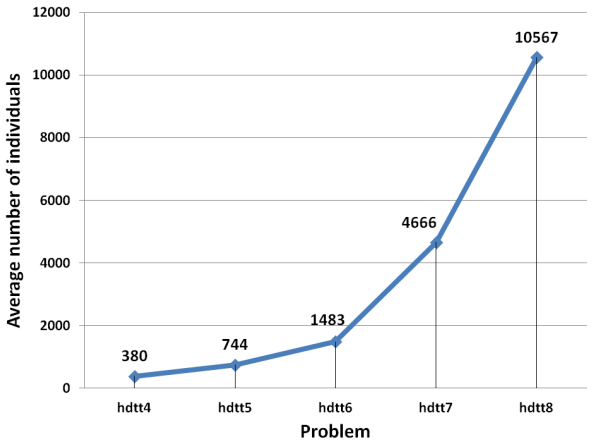

Fig. 6. The average number of individuals for the reproduction phase.

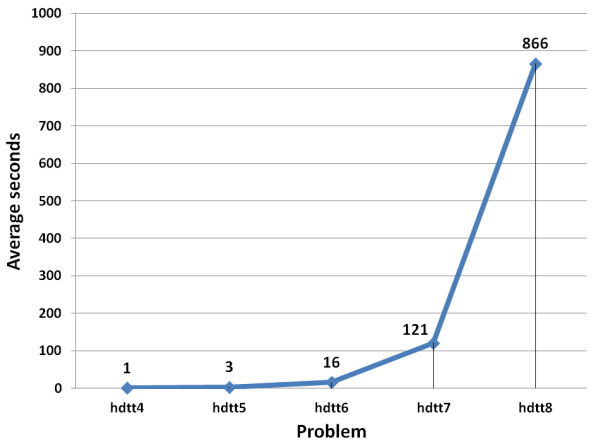

Fig. 7. The average CPU seconds over 20 runs for the proposed algorithm.

Fig.6 demonstrates the average number of individuals that the reproduction phase of proposed algorithm requires to get the final timetable. Fig.7 shows the average CPU processing time of the proposed algorithm, and how the proposed algorithm is fast to generate the final timetable.

Table 2 compares the performance of the proposed algorithm and the other methods. The first column indicates the best cost result achieved, and the avg cost column for the average cost of all runs. The results show that the proposed algorithm yields a zero cost value with accepts processing time for all instances. Also, as can be seen from Table 2 above, the performance of the proposed algorithm is competitive when compared to the other methods, and performs the best finding timetables for all 20 runs conducted for each data set.

6. CONCLUSION

The proposed algorithm combining Genetic Algorithm (GA) with Hill Climbing Optimization (HCO) is described for solving CTTP. The proposed algorithm is based on the mutation operation of GA only. The RPD is used to discover the fittest mutation function of 16 mutation functions. The combination of GA and HCO allow to decrease the number of generation which produced by GA. That is shown in all artificial hdtt instances, the proposed algorithm gives optimal solutions with one hundred percentages of fitness function values within a few seconds. The experiment were design to show the effectiveness of the proposed algorithm over other approaches in literature. That insure the proposed algorithm can be added as a new method for solving CTTPs. The future work will investigate to solve other timetable problems such as examinations, transportation and sporting scheduling. The performance of the proposed algorithm can be improved by release the hill climbing rate be changeable during the production phase to wide the search space, which will enhance the quality of the obtained results.

7. REFERENCES

[1] D Abramson and H Dang. School timetables: A case study in simulated annealing. In *Applied simulated annealing*, pages 103–124. Springer, 1993.

[2] Ammar M. Ammar, Adel S. Elmaghraby, Mervat H. Gheith, and Hesham A. Hassan. Multi-agent system for solving scheduling problem. Master's thesis, Department of Computer Science and Information, Institute of Statistical Studies and Research, Cairo University, 2005.

[3] Rakesh P Badoni and DK Gupta. A hybrid algorithm for university course timetabling problem. *Innovative Systems Design and Engineering*, 6(2):60–66, 2015.

[4] Asaju Laaro Bolaji, Ahamad Tajudin Khader, Mohammed Azmi Al-Betar, and Mohammed A Awadallah.

University course timetabling using hybridized artificial bee colony with hill climbing optimizer. *Journal of Computational Science*, 5(5):809–818, 2014.

[5] Marco P Carrasco and Margarida Vaz Pato. A comparison of discrete and continuous neural network approaches to solve the class/teacher timetabling problem. *European Journal of Operational Research*, 153(1):65–79, 2004.

[6] Ruey-Maw Chen and Hsiao-Fang Shih. Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms*, 6(2):227–244, 2013.

[7] M Doulaty, MR Feizi Derakhshi, and M Abdi. Timetabling: A state-of-the-art evolutionary approach. *International Journal of Machine Learning and Computing*, 3(3):255–258, 2013.

[8] Mahmoud M El-Sherbiny and Yasser M Ibrahim. An artificial immune algorithm with alternative mutation methods: applied to the student project assignment problem. In *International conference on innovation and information management (ICIIM2012), Chengdu, China*, volume 36, pages 149–158, 2012.

[9] Mahmoud Moustafa El-Sherbiny. Alternate mutation based artificial immune algorithm for step fixed charge transportation problem. *Egyptian Informatics Journal*, 13(2):123–134, 2012.

[10] Cheng Weng Fong, Hishammuddin Asmuni, Barry McCollum, Paul McMullan, and Sigeru Omatu. A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems. *Information Sciences*, 283:1–21, 2014.

[11] Michel Gendreau and Jean-Yves Potvin. Metaheuristics in combinatorial optimization. *Annals of Operations Research*, 140(1):189–213, 2005.

[12] Zhipeng Lü and Jin-Kao Hao. Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 200(1):235–244, 2010.

[13] Alade O Modupe, Omidiora E Olusayo, and Olabiyisi S Olatunde. Development of a university lecture timetable using modified genetic algorithms approach. *International Journal*, 4(9):163–168, 2014.

[14] Antony E Phillips, Hamish Waterer, Matthias Ehrgott, and David M Ryan. Integer programming methods for large-scale practical classroom assignment problems. *Computers & Operations Research*, 53:42–53, 2015.

[15] Michael Pimmer and Günther R Raidl. A timeslot-filling heuristic approach to construct high-school timetables. In *Advances in Metaheuristics*, pages 143–157. Springer, 2013.

[16] Rushil Raghavjee and Nelishia Pillay. A comparison of genetic algorithms and genetic programming in solving the school timetabling problem. In *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, pages 98–103. IEEE, 2012.

[17] Marcus Randall and David Abramson. A general metaheuristic based solver for combinatorial optimisation problems. *Computational optimization and applications*, 20(2):185–210, 2001.

[18] Khalid Shaker and Salwani Abdullah. Incorporating great deluge approach with kempe chain neighbourhood structure for curriculum-based course timetabling problems. In *Data Mining and Optimization, 2009. DMO'09. 2nd Conference on*, pages 149–153. IEEE, 2009.

[19] Kate A Smith, David Abramson, and David Duke. Hopfield neural networks for timetabling: formulations, methods, and comparative results. *Computers & industrial engineering*, 44(2):283–305, 2003.

[20] Defu Zhang, Yongkai Liu, Rym MHallah, and Stephen CH Leung. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 203(3):550–558, 2010.