International Journal of the Physical Sciences Vol. 5(16), pp. 2516-2522, 4 December, 2010
Available online at http://www.academicjournals.org/IJPS
ISSN 1992 - 1950 ©2010 Academic Journals

Full Length Research Paper

Adaptive randomized descent algorithm for solving course timetabling problems

Anmar Abuhamdah* and Masri Ayob

Data Mining and Optimization Group (DMO), Centre of Artificial Intelligence Technology (CAIT) Faculty of Computer Science, Universiti Kebangsaan Malaysia, Selangor, 43600, UKM, Selangor, Malaysia.

Accepted 1 December, 2010

This work proposes an Adaptive Randomized Descent Algorithm (ARDA) for solving university course timetabling problems. The work aims is to produce an adaptive algorithm that can produce good quality timetable by assigning a set of courses (events) and students to a fixed number of timeslots and rooms subject to a set of constraints. ARDA delays the comparison between the quality of the candidate solution and the current solution. ARDA use a threshold value (that is calculated based on the average quality of some recently accepted solution) as an acceptance criterion. ARDA can adaptively manage to escape from local optima by intelligently changing the threshold value when the search is trap in local optima. This is done by estimating an appropriate threshold value based on the history of the search. Results tested on the Socha's benchmark datasets showed that, ARDA produces significantly good quality solutions when compared with late acceptance strategy in hill climbing, average late randomized descent within a reasonable time and comparable to other approaches tested on Socha's dataset.

Key words: Course timetabling problems, late acceptance strategy hill climbing.

INTRODUCTION

Generally, university course timetabling problems involves assigning a set of courses (events), teachers and students to a fixed number of timeslots and rooms (Petrovic and Burke, 2004) subject to variety of constrains. Constraints in a timetabling problem can be classified as hard and soft constraints (Petrovic and Burke, 2004). The goal of timetabling is to satisfy all hard constraints and attempt to accommodate the soft constraints as much as possible (for producing a high-quality timetable). A feasible timetable must satisfy all hard constraints, whilst soft constraints can be violated if necessary, but each violation of the soft constraints will be penalized. A smaller penalty value indicates a good quality timetable. University course timetabling problem has been classified as an NP-hard problem, to which it is difficult to find an optimal solution within a reasonable time (Schaerf, 1995). Finding good-quality solutions to these problems depends on the technique itself and the structure employed during the search (Schaerf, 1995).

In the last few years, a lot of approaches had been applied to solve university course timetabling problems. Some of these approaches are great deluge (Burke et al., 2003), simulated annealing (Elmohamed et al., 1998) tabu search (Costa, 1994) and randomized descent method (Schaerf, 1999). For further information on previous works, please refer to the following survey/overview papers (Schaerf, 1995; De Werra, 1985). Recently, a new approach on timetabling problems based on basic local search had been proposed by Burke and Bykov (2008). In our previous work (Abuhamdah and Ayob, 2010) we extended the work in Burke and Bykov

*Corresponding author. E-mail: anmar@ftsm.ukm.my. Tel: 0060173668114.

Table 1. Eleven datasets (Socha et al., 2002; Chiarandini et al., 2006).

Dataset	No. of students	No. of events	No. of rooms	No. of features	ApproxF/R	F usage (%)	CD
S*1*	80	100	5	5	3	0.7	10.96
S*2*	80	100	5	5	3	0.7	13.92
S*3*	80	100	5	5	3	0.7	9.71
S*4*	80	100	5	5	3	0.7	7.16
S*5*	80	100	5	5	3	0.7	15.10
M1	200	400	10	5	3	0.8	37.38
M2	200	400	10	5	3	0.8	37.66
M3	200	400	10	5	3	0.8	40.44
M4	200	400	10	5	3	0.8	37.50
M5	200	400	10	5	3	0.8	28.27
L	400	400	10	10	5	0.9	45.57

(2008), by introducing average late acceptance randomized descent(ALARD), to solve university course timetabling problem. However, both LAHC and ALARD are descent heuristic. The disadvantage of the descent heuristic techniques is that, they are incapable of escaping local optima (minima) (Schaerf, 1999).

Therefore in this work, we propose an adaptive average late acceptance randomized descent (ARDA), a new local search heuristic (which is not descent heuristic) based on randomized descent method (Schaerf, 1999) that extends our previous work (Abuhamdah and Ayob, 2010). The aim of this study is to investigate the performance of applying the adaptive randomized descent algorithm for solving university course timetabling problems. In order to evaluate the effectiveness of the ARDA, we made a comparison between the performance of ARDA, LAHC (Burke and Bykov, 2008), ALARD (Abuhamdah and Ayob, 2010) and other approaches which were tested over Socha's university course timetabling datasets (Socha et al., 2002).

Problem description

In this work, eleven datasets instances introduced by Socha et al. (2002) were used, which only tackle the students' satisfaction for the university course timetabling problem. The problem consists of: A set of rooms (R) in which events can take place; a set of events (courses) (E) to be scheduled in 45 timeslots (5 days of 9 h each with one hour for each timeslot); a set of features (F) characterizing the rooms and required by events; a set of students (S) who attend the events. These datasets are categorized into three groups, small (S*1, S2, S3, S4 and S5*), medium (M1, M2, M3, M4 and M5) and large (L) as shown in Table 1 (Socha et al., 2002). Table 1 also shows, the conflict density (CD) for each dataset (representing the complexity which is calculated as in Chiarandini et al. (2006).

These datasets were collected from various real-world university course timetabling problems (Socha et al., 2002). The problem consists of three hard constraints (Hc1, Hc2 and Hc3) and three soft constraints *(Sc1, Sc2, and Sc3*) as follows: Hc1, No student can be assigned to more than one event at the same time; Hc2, the room capacity must be greater than or equal to all the attending students and satisfy all the features required by the event that assigned to it; Hc3, no more than one event can take place in each room at same timeslot; *Sc1*, a student should not have an event in the last time slot of the day; *Sc2*, a student should not have more than two events consecutively. *Sc3*, a student should not have a single event on a day.

As in Socha et al. (2002), we measure the quality of timetable by penalizing each violation of the soft or hard constraints by '1' for each student who is involved in this situation. The quality of the candidate solution *S*,* f (*S**), is simply calculated as the summation between the hard constraints violations hvc (total penalty of hvc) of *S** for all students and soft constraints violations svc (total penalty of svc) of *S** for all students (*S* equal total number of students), as shown in Equation (1) by Rossi-Doria et al. (2003).

$$f(S^*) = \sum_{i=1}^{S} hvc(S^*) + \sum_{i=1}^{S} svc(S^*) \quad (1)$$

Neighborhood structures

In this work, we use composite (Grabowski and Pempera, 2000) neighborhood structures with different five

neighborhood structures (*NS1, NS2, NS3, NS4, and NS5*) for solving the university course timetabling problem as in Abuhamdah and Ayob (2010). These are: *NS1,* Randomly select two courses and swap their timeslots if feasible (and swap their rooms if necessary); *NS2*, Randomly select two timeslots and simply swap all the courses in one timeslot with all the courses in the other timeslot; *NS3,* Randomly select four courses and swap their timeslots if feasible (and rooms if necessary); *NS4,* Randomly select a course, feasible timeslot and feasible room and move the course to the new timeslot (and move the course to the new room if necessary); *NS5,* Choose the highest penalty courses from a random 15% selection of the courses and assign to random feasible timeslots.

The Adaptive Randomized Descent Algorithm (ARDA)

The adaptive average late acceptance randomized descent is an improvement heuristic based on late acceptance strategy in hill climbing (LAHC) which was proposed by Burke and Bykov (2008). The idea of LAHC is to delay the comparison between the qualities of the candidate solution with a solution quality, which was "current" several steps before the current solution. This is done by maintaining a list of accepted solution's quality (value of the penalty cost).

LAHC allows some worsening moves and it have a single parameter (the list length) (Burke and Bykov, 2008), whilst other algorithm such as simulated annealing and great deluge have a function (Burke and Bykov, 2008).The list in LAHC, is used as the acceptance criterion (Burke and Bykov, 2008). LAHC is free from drawback except need to investigate the suitable length of the list. Whilst, other algorithm such as tabu search (when we increase of the length of the tabu list) always causes higher computational expense (Burke and Bykov, 2008). In LAHC, the search can maintain a list of any length without extra expense (Burke and Bykov, 2008). On the other hand, the limitation of LAHC is to decide a suitable length of list L for each different benchmark problem (Burke and Bykov, 2008).

Therefore, in our previous work, we proposed the average late acceptance randomized descent (ALARD) (Abuhamdah and Ayob, 2010), which is based on LAHC idea, but use an average value of the values in a list of accepted solution as the acceptance criterion (Abuhamdah and Ayob, 2010). However, both LAHC and ALARD are descent heuristics (where they only accept solutions that are better than or equal quality of the selected solution or average quality, respectively). Thus, they can easily be trapped in local optima. In respect to this, we propose (a non descent heuristic) the adaptive average late acceptance randomized descent (ARDA, which enhance the ALARD approach) to overcome the weakness in LAHC and ALARD.

As in ALARD, ARDA also use a threshold value as an acceptance criterion. Apart from that, ARDA can adaptively manage to escape from local optima by adding the estimated value to the values in list L, which increases the threshold value. That is, we allow some slightly worse solution to be accepted which might help the algorithm to escape from a local optima. The following shows the pseudo code for ARDA to solve university course timetabling problems.

In this work, we used a constructive heuristic that was originally proposed by Landa-Silva and Obit (2008) to produce an initial solution and we implemented five different neighborhood structures (NS1, NS2, NS3, NS and NS5) as in Abuhamdah and Ayob (2010), which are simultaneous with Grabowski and Pempera (2000).

ARDA pseudo code

Given an initial solution, S_o and its quality, as $f(S_o)$; Let S_{best}, be the best obtained solution, $f(S_{best})$, be the quality of S_{best} ; S^*, be the candidate solution, $f(S^*)$ be the quality of S^*; L, be the list of accepted solutions quality of the length ARD; Avg, be the average of the values in L; N, be the maximum number of iterations; n be the number of iterations; IT, be the maximum number of idle IT; EV, be the estimated value; Idle be the idle Iterations; arrayc, be the counter for the length of the array that stored EV. The pseudo code for ARDA to solve university course timetabling problems can be seen in Figure 1.

Figure 1 shows that, our approach starts with a given initial solution S_o and quality of S_o, as $f(S_o)$. Let S_{best} be the best obtained solution, $f(S_{best})$, be the quality of S_{best} ; S^*, be the candidate solution, $f(S^*)$, be the quality of S^*; L_{Length}, be the length of the list L; *Avg,* be the average of the values in L; *EV,* be the estimated value stored in the list A; *Idle,* be the ARDA idle Iterations. ARDA-RR starts by initializing the required parameters (Step 1, Figure 1) by assigning all elements in L equal to the quality of the initial solution $f(S_o)$ and set the average value, *Avg* equal to $f(S_o)$. In this work, we use a fix size of L (L size = 20). In the improvement phase (Step2), we iteratively improve the initial solution S_o until the stopping criteria is met. At this phase, we generate some neighbour solutions (in our case, randomly generate some "5" feasible neighbours from each neighborhood of the five neighbourhoods (*NS1-NS5*)) and the best feasible neighbour (S^*) will be selected to be the candidate solution (Step 2.1).

At each iteration, if the candidate solution $S^{*\ better}$ is better than the best solution S_{best}, then we will accept the solution S^* (Step 2.2, that is $S_o = S^*$), update the frequency of *EV* in the list A, update $f(S_{best}) = f(S^*)$ and sort the element in descending order based on their frequency or if $f(S^*)$ is just better than *Avg*, we will only

```
Step 1: Initialization phase
    Determine initial candidate solution So and f(So)
    Set idle = 0;  Sbest=So;   f(Sbest)= f(So);
    Set L Values = f(So);  Avg=average of L values;
Step 2: Improvement (Iterative) Phase
    while termination condition is not satisfied :
  Step 2.1: Generate some neighbor solutions
        and choose the best Neighbor S*;
  Step 2.2: Accepting Solution
    if f(S*) < f(Sbest)
    Idle =0;  EV= f(So) - f(S*);
    Sbest:=S*;      f(Sbest)=f(So);  So=S*;
     Update the frequency of EV in the A (or
      add EV in A if it is not exist).
       Sort the EV elements in A in descending
       order based on their frequency.
     elseif f(S*) < Avg  // good solution
           Replace the element value that was the
        longest time in L with f(S*);
     Idle =0;  So= S*;  Recalculate Avg;
 else  // bad solution
     So= Sbest;      Idle = Idle + 1;
     end if
  Step 2.3: Idle Iteration
     if Idle >= Maximum number of Idle iterations
     EV = A [0]; // get the first element in A
      Add all values in L with EV;
          Rotate left all elements in A;
     Recalculate Avg;
     end if
     end while
Step 3: Termination phase
    Return the best found solution  Sbest
```

Figure 1. ARDA pseudo code.

accept $f(S^*)$ and replace the element value that was the longest time in *L* with the quality of the new accepted solution. We then recalculate *Avg*, otherwise, the S^* will be rejected. We will then, increase the idle iteration by one, updating $S_o = S_{best}$ and proceed with the next iteration. In Step 2.3, we set the estimated value, *EV* equals to the first value in *A* (the one with the highest marked of repetition), when number of idle iterations equals to the maximum number of idle, then, we update all values in *L* (by adding *EV* to the values), update all values in *A* (by rotate left all elements in *A*) and recalculate *Avg*.

In the termination phase Step3, we return the best solution found S_{best} , after we iteratively improving the initial solution S_o until the stopping condition is met (Step 2). The main differences between ARDA, LAHC and ALARD are as follows: (1) ARDA and ALARD use an average value of the values in a list of accepted solution as the acceptance criterion, whilst in LAHC; the acceptance criterion is based on the one of the selected accepted solution cost values; (2) ARDA and ALARD differs from the basic LAHC as it uses randomized descent method (which evaluate more possible neighbors) that, may lead to better improvement instead of using simple hill climbing; ARDA can adaptively attempts' to escape from local optima by intelligently changing the threshold value when the search trap in local optima (when the quality of the generated solutions from some neighborhoods are not better than the best obtained solution quality). This is done by estimating an appropriate threshold value based on history of search (by counting the number of idle iteration and using the counter for

Table 2. Comparison between our methodology and other local hybrid meta-heuristic search results on course timetabling problem.

Dataset	ARDA		R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
	Min	Avg												
S1	0	0.73	0	0	0	0	0	0	0	3	0	0	0	0
S2	0	0.82	0	0	0	0	0	0	0	4	0	0	0	0
S3	0	1.55	0	0	0	0	0	0	0	6	0	0	0	0
S4	0	1.45	0	0	0	0	0	0	0	6	0	0	0	0
S5	0	0.55	0	0	0	0	0	0	0	0	0	0	0	0
M1	82	96.73	77	80	93	143	153	338	242	140	168	221	55	71
M2	78	93.45	73	105	98	130	137	326	161	130	160	147	70	82
M3	136	151.45	133	139	149	183	207	384	265	189	176	246	102	137
M4	73	93.36	69	88	103	133	142	299	181	112	144	165	32	55
M5	103	120.73	101	88	98	169	186	307	151	141	71	130	61	106
L	680	700.72	627	730	680	825	863	-	-	876	417	529	653	777

R1, HPCA (Abuhamdah and Ayob, 2009); R2, EGD (McMullan, 2008); R3, DSSA (Abdullah et al., 2010); R4, ALARD (Abuhamdah and Ayob, 2010); R5, LAHC (Abuhamdah and Ayob, 2010); R6, VNS-TS (Abdullah et al., 2005); R7, RII-CN (Abdullah et al., 2007); R8, NLGD (Obit et al., 2009); R9, MHSA (Al-Betar et al., 2010); R10, HEA (Abdullah et al., 2007); R11, TB-MA (Turabieh and Abdullah, 2009); R12, NLGDHH-SM (Obit et al., 2009).

improvement rate, *EV*). Whereas, the accepting criterion for LAHC and ALARD are static, which may easily be trapped in local optima.

RESULTS

In this work, the results of our algorithm were obtained out of 11 runs and 200,000 iterations and tested on a PC with an Intel dual core 1800 MHz, 1GB RAM. Table 2 shows the comparison between our ARDA approaches LAHC, ALARD and other local hybrid meta-heuristic searches that were tested on Socha benchmark datasets. The best results are presented in bold. Results in Table 2 show that, ARDA outperformed LAHC (R5) and ALARD (R4) in all datasets and are comparable to other results. For small datasets, we obtain our results within 3 to 10 min, whilst, for medium and large datasets, we took between 4 to 9 h to achieve the results. Results in Table 2 also show that, among these approaches, Turabieh and Abdullah (2009) in R11, which applied incorporating Tabu search into memetic approach for enrolment-based university course timetabling problems; have outperformed many other approaches in the literature (with regards to Socha benchmark datasets).

Whereas Al-Betar et al. (2010) in R9, applied a harmony search with multi-pitch adjusting rate for the university course timetabling. They also obtained optimal solution for all small datasets and the result for all medium datasets obtained within the range of published results. Meanwhile, they obtained the best result for medium 5 dataset. Meanwhile, the observation of the results obtained indicates that, ARDA performed well and obtained a good quality solution for all small, medium and large datasets. The results of all small datasets scored zero, the same as some of the published results; medium and large datasets were within the range of the published results. The results in Table 2 also shows that, ARDA obtained better solutions than single based local search results and obtained a very good-quality solution.

Figure 2 shows the box and whisker plot details of the ARDA results for 11 runs (Figure 2) shows, the box and whisker plot that summarize the results of 11 runs for each dataset by ARDA algorithm on Socha benchmark datasets. For all small (except small 4) and medium 2 datasets, we can see that, most of the runs produced high quality solution that are close to the best. This indicates that, the algorithm is stable and consistent and most of the time can produce very good quality solution. The result also shows that, ARDA is capable of producing high quality solutions for all datasets and are comparable with the best-known results obtained in literature. Table 3 present details analysis of the ARDA algorithms on the Socha benchmark datasets. Results in Table 3 show that, ARDA performance is consistent in producing good quality results indicated by smaller values in the standard deviation. As our algorithm can produce good quality solution in the same range of time when compared with the result obtained by other approaches in Table 2.

DISCUSSION

This work had proposed a new heuristic search called an

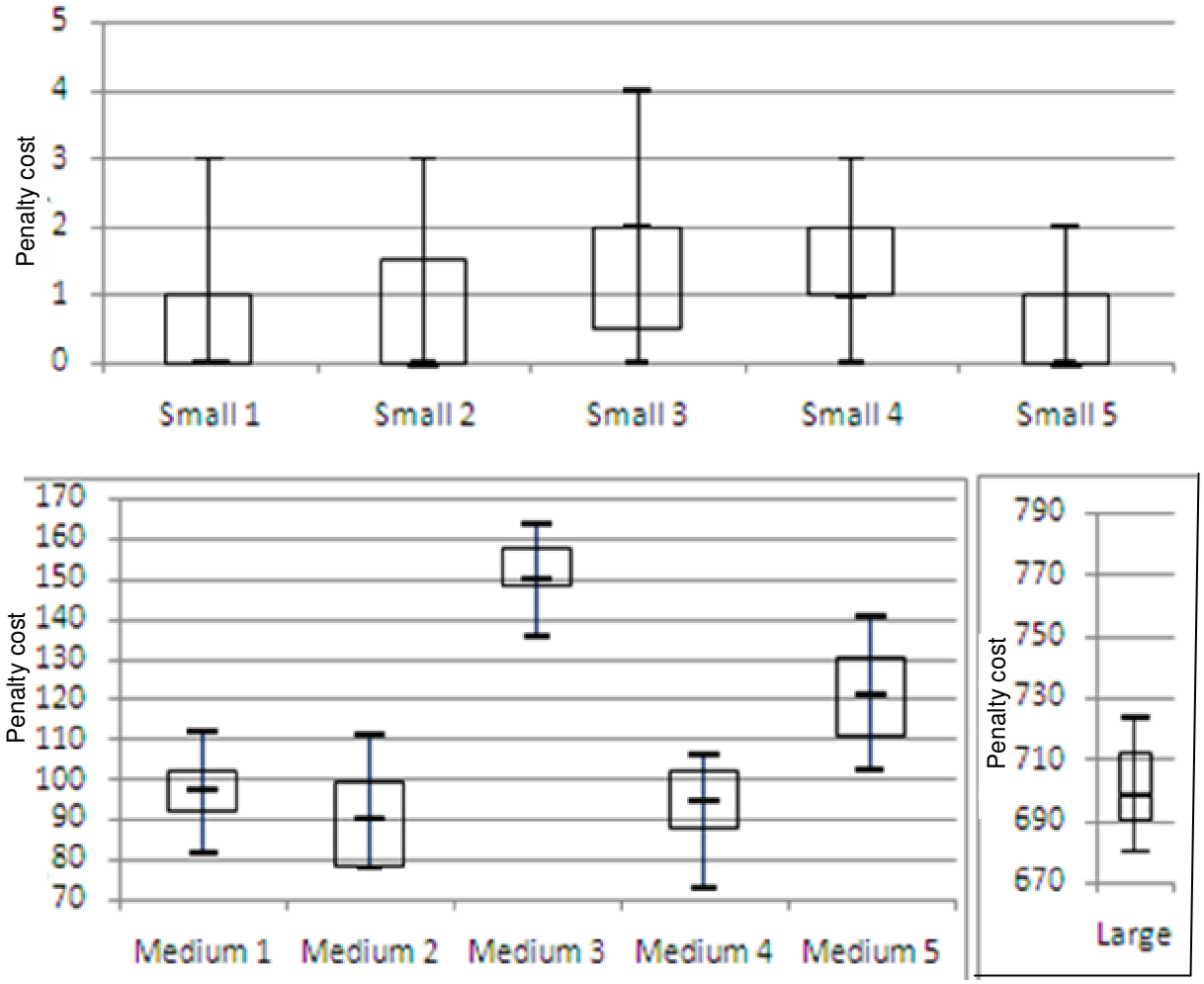

Figure 2. Box and whisker plot of ARDA for all datasets.

Table 3. Statistical analysis of our algorithm (ARDA) applied to Socha benchmark datasets.

Dataset	fmin	favg	Std. dev.(σ)	Iter	Time/s
s*1*	0	0.73	1.01	8600	166
s*2*	0	0.82	1.08	9537	189
s*3*	0	1.55	1.29	13655	267
s*4*	0	1.45	1.04	11694	228
s*5*	0	0.55	0.69	8694	159
m1	82	96.73	8.34	120350	21514
m2	78	93.45	10.03	107638	19568
m3	136	151.45	7.80	149365	23569
m4	73	93.36	10.59	165486	27522
m5	103	120.73	12.80	154620	25656
l	680	700.72	14.49	117620	28501

fmin, is the best score out of 11 runs; favg, the average score for all the 11 runs; σ, the standard deviation for all the 11 runs; Iter, total number of iteration proceeding to find that best score solution; Time/s, total CPU time on our computer needed to find the best score solution.

adaptive randomized descent algorithm (ARDA) to solve university course timetabling problems. Our works is an extension to average late acceptance randomized descent (ALARD), which is based on late acceptance strategy hill climbing (LAHC) that was proposed by Burke and Bykov (2008). The idea of ARDA, ALARD and LAHC are to delay the comparison between the quality of the candidate solution and the current solution. ARDA and ALARD use a threshold value as an acceptance criterion, whilst, LAHC use a selected solution's quality from a list of recently accepted solution's quality as acceptance criterion. ARDA can adaptively manage to escape from local optima by intelligently increasing the threshold when the search is trapped in local optima, which is done by estimating appropriate threshold values based on history of search. Whilst, LAHC and ALARD do not have this capability.

In order to evaluate the effectiveness of ARDA for solving the university course timetabling problem, we test ARDA on Socha benchmark dataset (Socha et al., 2002). Results indicated that, the performance of ARDA approach is comparable to the other approaches in the literature and capable of producing good-quality solutions in reasonable time. Our future work will investigate on how to intelligently manage neighborhood structures in order to enhance the performance of ARDA.

REFERENCES

Abdullah S, Burke EK, McCollum B (2005). An Investigation of Variable Neighbourhood Search for Course Timetabling". In: The Proceedings of the 2nd nnnMultidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, July 18th-21st, p. 413-427.

Abdullah S, Burke EK, Mccollum B (2007). A hybrid evolutionary approach to the university course timetabling problem. IEEE Congres on Evolutionary Computation, Singapore, p. 1764-1768.

Abdullah S, Burke EK, McCollum B (2007). Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem. In Proceedings of MIC05: The 6th Metaheuristic International Conference, Vienna, Austria, In Metaheuristics - Progress in Complex Systems Optimization, Computer Science Interfaces Book Series, Springer Operations Research, ISBN-13:978-0-387-71919-1, 39: 153-169.

Abdullah S, Shaker K, McCollum B, McMullan P (2010). Dual Sequence Simulated Annealing with Round-Robin Approach for University Course Timetabling. In: Evolutionary Computation in Combinatorial Optimization, LNCS, Volume 6022, Springer Berlin, Heidelberg, 1-10.

Abuhamdah A, Ayob M (2009). Hybridization Multi-Neighbourhood Particle Collision Algorithm and Great Deluge for Solving Course Timetabling Problems. Proceeding in 2009 2nd Conference On Data Mining and Optimization, 27-28 October 2009, Selangor, Malaysia, IEEE, 108-114, October.

Abuhamdah A, Ayob M (2010). Average Late Acceptance Randomized Descent Algorithm for Solving Course Timetabling Problems. Proceeding in 4th International Symposium on Information Technology, Selangor, Malaysia, IEEE, 2(15-17): 748-753.

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

Burke EK, Bykov Y (2008). A Late Acceptance Strategy in Hill-Climbing for Exam Timetabling Problems. in the Proceeding PATAT '08, Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, Universit de Montral, Montreal, Canada, August.

Burke EK, Bykov Y Newall J, Petrovic S (2003). A time-predefined approach to course timetabling. Yugoslav J. Operations Res., (YUJOR), 13(2): 139-151.

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

Costa D (1994). A tabu search for computing an operational timetable. Eur. J. Oper. Res., 76: 98-110.

De Werra D (1985). An Introduction to Timetabling. Eur. J. Oper. Res., 19: 151-162.

Elmohamed MAS, Coddington P, Fox G (1998). A comparison of annealing techniques for academic course scheduling. Selected Papers from 2nd International Conference on the Practice and Theory of Automated Timetabling (PATAT II), Toronto, Canada, Lecture Notes in Computer Science 1408, Springer-Verlag., pp. 92-112.

Grabowski J, Pempera J (2000). Sequencing of jobs in some production system". Theory and methodology. Eur. J. Oper. Res., 125: 535-550.

Landa-Silva D, Obit JH (2008).Great Deluge with Non-linear Decay Rate for Course timetabling Problems, 2008 4th International IEEE Conference Intelligent Systems, 978-1-4244-1739-1/08/.

McMullan P (2008). An Extended Implementation of the Great Deluge Algorithm for Course Timetabling, ICCS International Conference of Computational Science, Part I, LNCS Lecture Note in Computer Science, vol. 4487: 538-545, Springer-Verlag Berlin Heidelberg, Germany.

Obit JH, Landa-Silva D, Ouelhadj D, Sevaux M (2009). Non-Linear Great Deluge with Learning Mechanism for Solving the Course Timetabling Problem. MIC 2009: The VIII Metaheuristics International Conference, Hamburg, Germany, pp.id1-id10.

Petrovic S, Burke EK (2004). University timetabling, Ch. 45 in the Handbook of Scheduling: Algorithms, Models, and Performance Analysis (eds. J. Leung), Chapman Hall/CRC Press.

Rossi-Doria O, Samples M, Birattari M, Chiarandini M, Dorigo M, Gambardella M, Knowels J, Manfrin M, Mastrolilli M, Paechter B, Paquete L, Stultzle T (2003). A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, Springer, Heidelberg, 2740: 329–354.

Schaerf A (1995). A Survey of Automated Timetabling, Technical Report CS-R9567, CWI, Amsterdam, NL.

Schaerf A (1999). Local Search Techniques for Large High-School Timetabling Problems. systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on Volume, 29(4): 368-377.

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 Computer Science, 2463(10): 1-13.

Turabieh H, Abdullah S (2009). Tabu based Memmetic approach, Incorporating Tabu Search into Memetic approach for enrolment-based course timetabling problems. (TS-MA), Proceeding in the 2nd Conference On Data Mining and Optimization, Selangor, Malaysia, IEEE, pp. 115-119.