Course Timetabling Using Forest Algorithm

Vahideh Sahargahi
M.Sc. in Artificial Intelligence, Faculty of Electronic and Computer, Tabriz University, Tabriz, Iran
Mohammadreza Feizi Derakhshi
PhD in Artificial Intelligenc, Faculty of Electronic and Computer, Tabriz University, Tabriz, Iran

Abstract
The timetabling problem is a subset of NP-Complete problems for which numerous solutions and studies have been presented. Algorithms provided for course timetabling use methods such as Particle Swarm Optimization (PSO), Memetic, fuzzy systems, Harmony Search approach and Ant Colony. The Forest algorithm is an optimization method proposed in recent years by researchers for optimization problems. In this article, it is used for course timetabling. Evaluation and comparison of the Forest algorithm with the Genetics Algorithm (GA) in 21 different databases showed its higher accuracy than GA. It also had a shorter runtime and achieved better results in less time.

Keywords:
Course timetabling, evolutionary algorithm, Forest algorithm

1. Introduction

Timetabling is one of the most important requirements of schools, training centers and universities. It is a subset of NP-Complete problems for which numerous solutions and studies have been presented.
Timetabling refers to assigning events to pre-defined time intervals subject to constraints for events. Constraints are divided into hard and soft types. Hard constraints should be satisfied for timetabling but soft constraints can be ignored, however, they must be applied as possible. The quality of timetabling is measured considering the satisfied soft constraints.
New approaches that are used for timetabling problems are evolutionary algorithms. Evolutionary algorithms are divided into several categories such as hill-climbing search, luminescent method and GA. The GA is one of the most powerful and widely used algorithms for search and optimization problems. The Forest algorithm is one of evolutionary methods recently proposed by researchers. The algorithm has not been used for timetabling. This paper uses the Forest algorithm for timetabling and evaluates its performance, and compares it with existing methods.
The article is organized as follows: Section 2 is literature review. Section 3 presents the Forest algorithm. Section 4 explains how to use the Forest algorithm for course timetabling and Section 5 presents evaluation conditions, evaluation results and comparison with GA. Conclusion and future works are presented in Section 5 of the thesis.

2. Literature review

Different optimization methods are used by different researchers for course timetabling. In [21] different methods for school timetabling and constraints considered in each method are provided. In [20] different methods school, university and examination timetabling and student classification are presented. Among methods used for course timetabling, one can point to the PSO algorithm [1][6][7], Memetic algorithm [2], fuzzy logic [3], harmony search [5] and Ant Colony [4][15][16].
The Hybrid Particle Swarm Optimization (HPSO) which is the improved version of PSO for course timetabling is presented in [1]. In this paper, soft constraints ignored in other articles are considered which one of the advantages for the proposed method is. It compares HPSO with GA based on the fitness function. Results indicate higher efficiency for HPSO.
In [6] the Constriction Particle Swarm Optimization (SPSO) method is proposed for course timetabling. With the exchange heuristic function, the method can form the course timetabling subject to the request of teachers, classroom and hard and soft constraints. The existence of local search for course timetabling solves the premature convergence problem of PSO while increasing the satisfaction of teachers and the quality of solutions.
In [7] GA is compared with PSO. Results show that PSO needs lower iterations than GA to achieve the optimal solution. The penalty of PSO is lower than GA.
The Memetic algorithm for course timetabling is proposed in [1]. One method that can be used to assess timetabling is to consider the number of unscheduled events. The use of this criterion as the fitness function can distinguish very similar solutions that are different in structure. Therefore, a different fitness function must be used to express the difference between solutions. The fitness function used in this article has features including the use of experience for learning, low calculation cost and considering the solution structure. When the number of samples is low or medium, this method has a higher efficiency compared to other

Manuscript received February 5, 2017
Manuscript revised February 20, 2017

heuristic methods including GA, H and GGA. The Memetic method produced better results, followed by the H, HSA and GGA algorithms, respectively.

The Ant Colony algorithm for course timetabling was proposed by Thepphakorn et al. in [4]. A new type of Colony algorithm called Best-Worst Ant System (BWAS) and Best-Worst Ant Colony System (BWACS) were added to the ACO program and then the local search strategy was added to the proposed methods in order to improve their efficiency and find the best solutions. Added local search increased the efficiency of proposed methods up to 40%, however it increased runtime.

The efficiency of BWAS and BWACS was more than ACO. About the two proposed methods, it can be said that BWACS is suitable for large-sized problems and BWAS for small-sized ones. In [11], Ayob et al. combined two different methods with Ant Colony for course timetabling. One hybrid technique is called ACS-TS, which combines Ant Colony and the Tabu Search algorithm, and the other is called ACS-SA which combines Ant Colony and Simulated Annealing algorithm. The evaluation results show that both hybrid approaches have higher efficiency and higher quality results compared to Ant Colony. The authors believe that the hybrid method can be used for other optimization problems.

In [15], Nothegger et al. proposed a method based on Ant Colony for course timetabling. The most important feature of the proposed method is the use of two separate but very simple pheromone matrices to improve convergence. They also used parallel program implementation and improved the quality of the proposed method. The ITC2007 database was used for evaluation. The evaluation results indicate optimal timetables for different sizes of the database by the proposed method. In the next step, local search was added to increase the efficiency of Ant Colony. Results showed even greater efficiency with local search.

Both Ant Colony and Simulated Annealing algorithms were investigated and compared for examinations timetabling by Chmait et al. According to the results, the Ant Colony runtime was shorter than the Simulated Annealing algorithm because the latter needs more time to explore and evaluate neighbors in various periods of the program. In terms of cost, in most cases Ant Colony had lower costs than the Simulated Annealing algorithm. In Ant Colony, with increasing number of ants, runtime increases but it has no severe impact on improving the quality of timetabling. Moreover, with excessive reduction of the number of ants, runtime decreased but timetabling was achieved at high cost [16].

The BSC method is one of the oldest methods for course timetabling. In this method, at some point, events are sorted based on the degree of difficulty. The method can be done in many ways depending on sorting criteria. In [3] fuzzy logic is used to sort events in the BSC method. The evaluation results showed that compared to individual methods, the fuzzy method provides higher quality results for course timetabling with the BSC method. Moreover, the number of rescheduling required by the algorithm to find the solution was calculated for different methods. According to the results, the fuzzy approach needs the minimum number of rescheduling per database. The advantages of fuzzy method for calculating the difficulty level of events in the BSC algorithm include high quality and high efficiency when dealing with large databases.

In [23] the fuzzy theory is used to increase the efficiency of ant population. The course timetabling is presented as a 2D matrix in which rows represent (rooms and timeframe) and columns represent events. It is displayed as graphs. The evaluation results indicate the high efficiency of the algorithm for complex problems with multiple constraints.

In [10], Azmi et al. proposed the Adapted Harmony Search Algorithm (AHSA) for course timetabling and a hybrid harmony search method for course timetabling. In this method, the hill-climbing algorithm was used to improve local search and the PSO algorithm to improve global search for increasing the quality of course timetabling.

AHSA and Hybrid Harmony Search Algorithm (HHSA) were evaluated on databases with different sizes. These methods were compared with 26 methods which used the same databases. AHSA found appropriate solutions for small-and-medium-sized databases. The proposed method was compared with the RII, RRLS, MMAS, VNS, GHH, FMHO, HEA, and THH methods, which produced better results than GHH and FMHO for small databases. For medium-sized databases, it had better results than VSN and some FMHO modes [5].

3. Forest algorithm

In nature, any element is destroyed after a period of time. Trees are no exception of this rule. To maintain the generation, trees use seeding [24] including seeds that fall around trees or seeds that are moved to other regions by animals, insects, wind, etc. In the process of nature, some trees live longer than others. The Forest optimization algorithm aims to find this type of trees (quasi-optimal solution). The Forest optimization algorithm consists of three main steps:

1. Local seeding of trees
2. Forest population control
3. Global seeding

Like any evolutionary algorithm, the initial population is produced for start. Each tree represents a solution for the problem. The algorithm flowchart can be seen below:

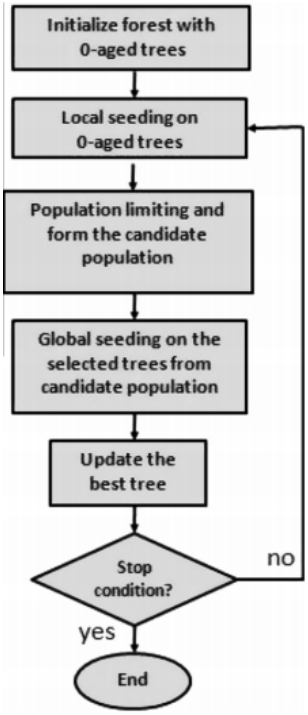


Figure 1: Flowchart of Forest optimization algorithm [24]

In the initial stage, the "Age" criterion is considered for each tree in addition to the problem variables. At the beginning of tree production, this value will be zero. In each iteration of local seeding, the Age value is incremented one unit (with the exception of newly produced trees). Each tree may live up to a certain age which is determined by the "life time" variable. If age exceeds life time, the tree will be removed from the cycle of nature.

During seeding, some seeds fall to the ground in the neighborhood of the tree, some of which turn into young trees. These trees compete to obtain resources such as sunlight, water, suitable soil and finally for survival. In the Forest optimization algorithm, this behavior is simulated in "local seeding". The number of trees that are produced locally is determined by the "local seeding changes" variable. This process is implemented on trees with zero age. Figure 2 shows the two stages of this process.

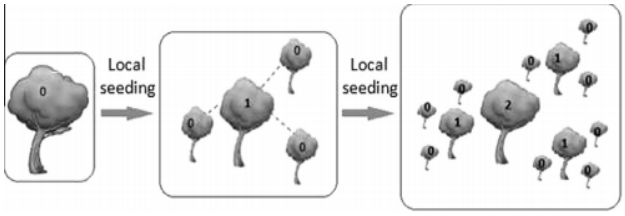


Figure 2: Local seeding [24]

After local seeding, the number of produced trees is controlled so that the forest population size does not become infinitely large. For this purpose, the two criteria of "life time" and "area limit" are considered. Trees with an age higher than life time are removed from the population and are added to the candidate population. Moreover, if the number of trees exceeds the forest size, after sorting by fitness, additional trees will be also added to the candidate population.

As was stated, some seeds are transferred to farther areas by animals, insects, wind, etc. which increase the tree's chance of generation survival. "Global seeding" tries to simulate this behavior. A percent of the candidate population is selected for this stage. The variable that determines this value is called "transfer rate". First, trees are selected based on transfer rates. Then a number of variables in each tree are randomly selected and the variable value is replaced by a random value within an appropriate range. By doing so, search is performed in other areas of state space. The newly generated tree is added to the population with zero age. The variables that are selected for change are determined by the "global seeding changes".

In the update section, the age of the best tree based on fitness value is changed to zero so that it is not removed due to aging while local seeding can produce optimal local solutions. The algorithm pseudo-code can be seen in Figure 3.

In order to simulate any problem by the algorithm, each solution can be displayed like Figure 4, so that if the problem has Nvar dimensions, each tree in the Forest will have Nvar+1 dimensions and Age is the tree age.

Algorithm FOA (life time, LSC, GSC, transfer rate, area limit)

Input: life time, LSC, GSC, transfer rate, area limit
Output: near optimal solution for objective function *f(x)*

1. Initialize forest with random trees
   - 1.1 Each tree is a (D+1)-dimensional vector x, x= (age, x_1, x_2, . . ., x_D) for a D-dimensional problem
   - 1.2 The "age" of each tree is initially zero
2. While stop condition is not satisfied do
   - 2.1 Perform local seeding on trees with age 0
     - For i=1: "LSC"
       - Randomly choose a variable of the selected tree
       - add a small amount dx- dx ε [-Δx, Δx]- to the randomly selected variable
     - Increase the age of all trees by 1 except for new generated trees in this stage
   - 2.2 Population limiting
     - Remove the trees with age bigger than "life time" parameter and add them to the candidate population
     - Sort trees according to their fitness value
     - Remove the extra trees that exceed the "area limit" parameter from the end of forest and add them to the candidate population
   - 2.3 Global seeding
     - Choose "transfer rate" percent of the candidate population
     - For each selected tree
       - Choose "GSC" variables of the selected tree randomly
       - Change the value of each variable with other randomly generated value in the variable's range and add a new tree with age 0 to the forest
   - 2.4 Update the best so far tree
     - Sort trees according to their fitness value
     - Set the age of the best tree to 0
3. Return the best tree as the result

Figure 3: Flowchart of Forest optimization algorithm [24]

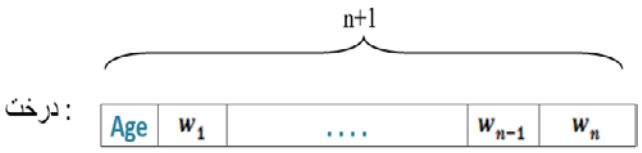


Figure 4: How to display a tree in the Forest optimization algorithm

4. Applying the Forest algorithm for course timetabling

The Forest algorithm includes various stages such as tree view, local seeding and global seeding that should be defined and applied according to the problem. Next, we explain how to use the algorithm for course timetabling.

4.1 Timetabling problem view and initialization

Each tree is considered as a time table. In the 2D view of time table, columns represent intervals and rows represent course locations. Figures 5 and 6 show the 1D and 2D views of each tree.

The coding method is permutation. In the permutation method, a tree is an integer n-member array and the numbers 1 through n are assigned to its different elements.

	Saturday 8-9	Sunday 9-10	…	Monday 13-14	…	Wednesday 17-18	Wednesday 18-19
Class 1	81	24	…	56	…	9	89
Class 2	76	12	…	53	…	39	69
…	29	59	…	20	…	86	75
Class n	18	42	…	15	…	33	5
Lab 1	89	9	…	96	…	72	49
…	41	30	…	78	…	57	86
Lab n	79	60	…	71	…	10	80

Figure 5: 2D view of course timetabling

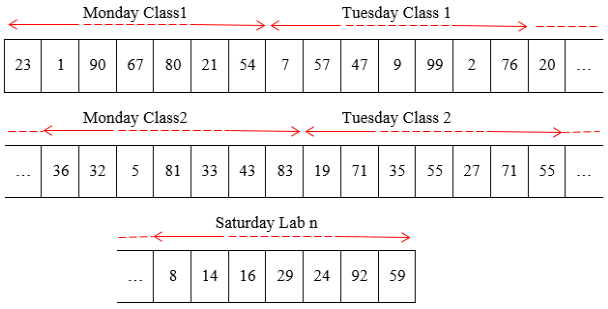


Figure 6: 1D view of course timetabling

Here, n is any course intended for timetabling. Since a classroom is not occupied in all time intervals, some cells of the table remain empty which will be filled with zero. In the proposed method, intervals are considered one hour.

4.2 Local seeding

Since course timetabling is a permutation problem, switching of the tree cells is used for local seeding. Thus, two cells from the trees with the zero age are randomly selected and switched. The resulting tree is a new tree that is added to the Forest population. The number of trees created from any tree in local seeding is equal to the numeric value of the "local seeding rate" parameter which is set by the user.

4.3 Global seeding

A percent of the candidate population is selected for this stage. First, trees are selected according to transfer rate. A number of cells equal to the "global seeding rate" value are randomly selected from each tree and are switched. The numerical value of the "global seeding rate" parameter is set by the user.

4.4 Constraints and evaluation function

Constraints are divided into hard and soft constraints. The evaluation function is equal to the weighted sum of hard and soft constraints. The equation of evaluation function is given below.

$$Cost = Alpha * \sum Hard_{Concentrate} + Beta * \sum Soft\_Concentrate \quad (1\text{-}4)$$

where Alpha and Beta parameters are the weighted sum of unsatisfied hard and soft constraints. In the course timetabling problem, the weight of hard constraints is selected much higher that soft ones. Constraints include the following:

- Hard constraints:
  - o Any course should not be held at specific times.
  - o Courses of a group in a single day must be held consecutively.
  - o Courses of a group should not be held at the same time.
  - o Each classroom has constraints for holding some courses and capacity (full or not).
- Soft constraints:
  - o Each classroom has a limited capacity.
  - o The number of days for a group has constraints.
  - o It is better that the units of a course are not held consecutively whenever possible.
  - o It is better that the units of a course are not held in the same classroom.

5. Evaluation of the proposed method and the results

This section presents the environments, evaluation conditions, the database used and the problem parameters. Then comparison of the proposed method with GA is presented. MATLAB R2010a was used for evaluation.

5.1 Database

The standard ITC2007 database was selected to evaluate the efficiency of the proposed method. The database contains 21 datasets with different sizes. The details are given in Table 1.

Table 1: ITC2007 database, examples used for competition of the 2008 methods

	Names	#Event	#Rooms	#Dayes	#Periods_per_day
Comp1	Fis0506-1	30	6	5	6
Comp2	Ing0203-2	82	16	5	5
Comp3	Ing0304-1	72	16	5	5
Comp4	Ing0405-3	79	18	5	5
Comp5	Let0405-1	54	9	6	6
Comp6	Ing0506-1	108	18	5	5
Comp7	Ing0607-2	131	20	5	5
Comp8	Ing0607-3	86	18	5	5
Comp9	Ing0304-3	76	18	5	5
Comp10	Ing0405-2	15	18	5	5
Comp11	Fis0506-2	30	5	5	9
Comp12	Let0506-2	88	11	6	6
Comp13	Ing0506-3	82	19	5	5
Comp14	Ing0708-1	85	17	5	5
Comp15	Ing0203-1	72	16	5	5
Comp16	Ing0607-1	108	20	5	5
Comp17	Ing0405-1	99	17	5	5
Comp18	Let0304-1	47	9	6	6
Comp19	Ing0203-3	74	16	5	5
Comp20	Ing0506-2	21	19	5	5
Comp21	Ing0304-2	94	18	5	5

5.2 Parameters initialization

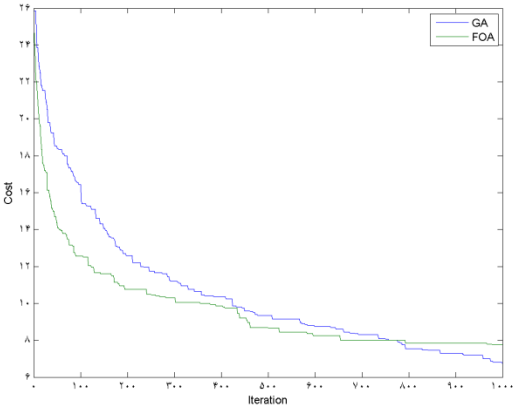
The algorithm parameters are shown in Table 2. Here, the m parameter is the number of course locations and n is intervals and E is the number of events. The values of these parameters are different depending on the dataset used. The numerical values of these parameters are shown in Table 1. The k parameter is constant.

5.3 Evaluation results

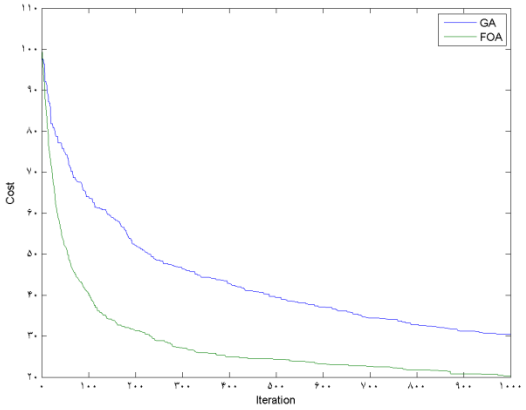
In this thesis, the proposed method was compared with GA considering different criteria. Figure 7: Continued ... shows the cost function of time table for the Forest and Genetics algorithms with standard databases in different conditions. According to the results, in more than 99.99% of iterations, the Forest algorithm has a lower cost than GA.

Table 2: Parameters of the Forest and Genetics algorithms

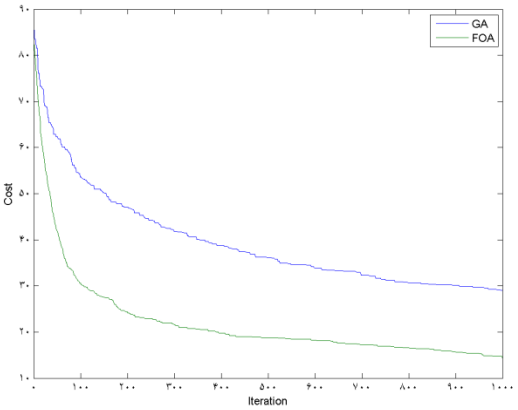
Algorithm	Parameters
GA	initial population = (kmn)/(E-1), k = 50, pc = 0.8, pm = 0.3, iteration = 1000
FOA	initial population = 40, life time = 20, ares limit = 40, transfer rate=0.3, iteration=1000, Dimension = mn, LSC = 0.3 of the Dimension, GSC = 0.015 of the Dimension



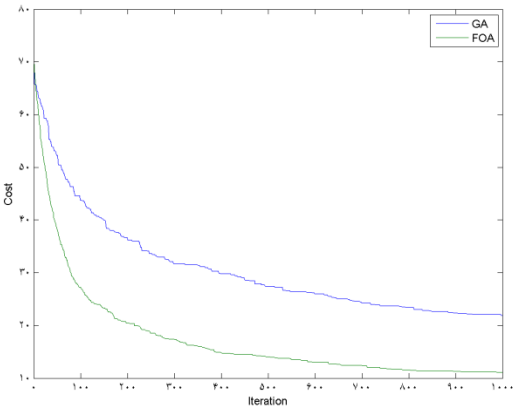
comp01



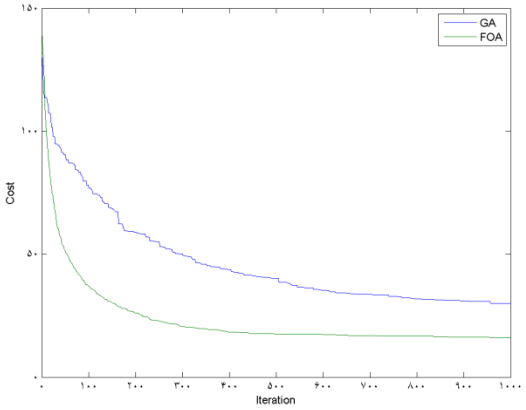
comp02



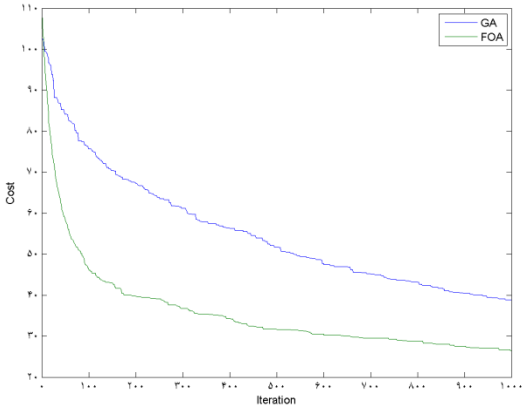
comp03



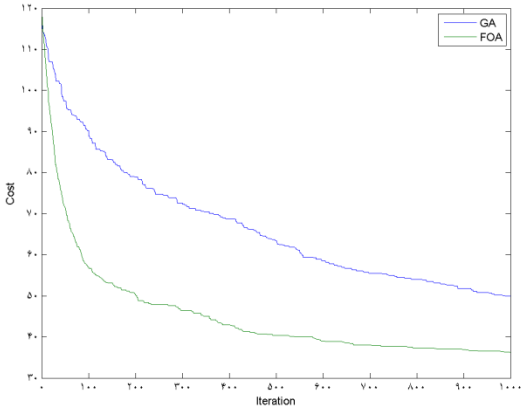
comp04



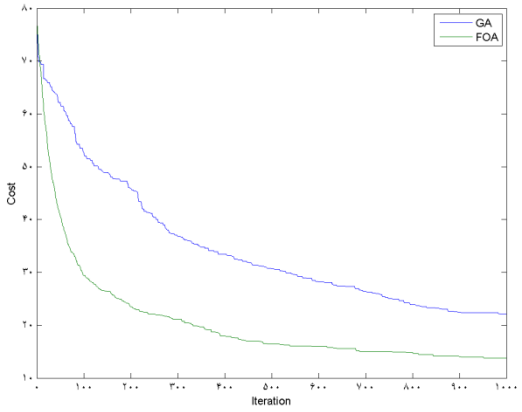
comp05



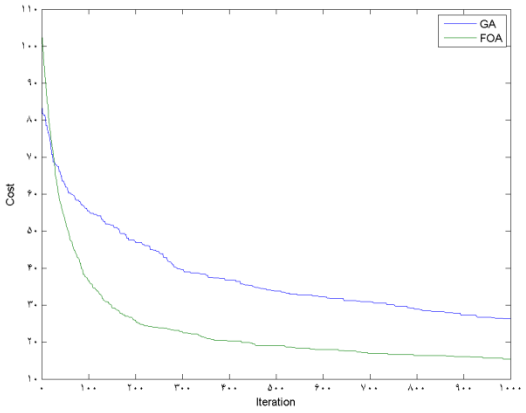
comp06



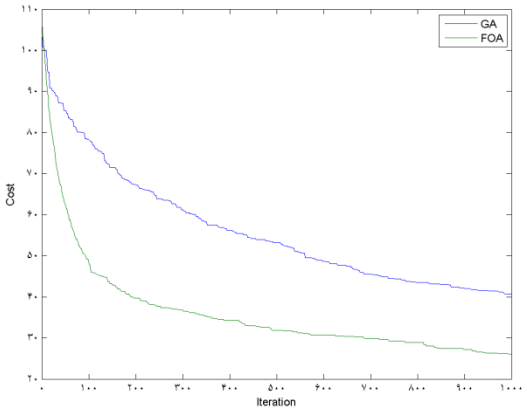
comp07



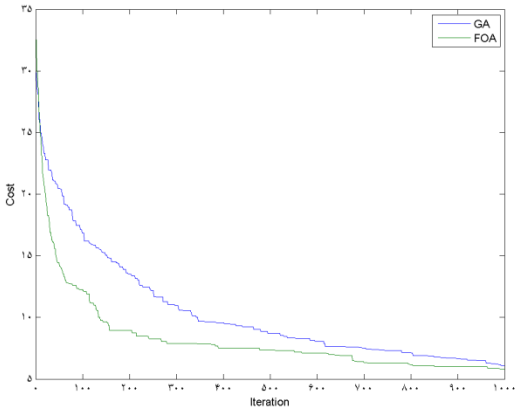
comp08



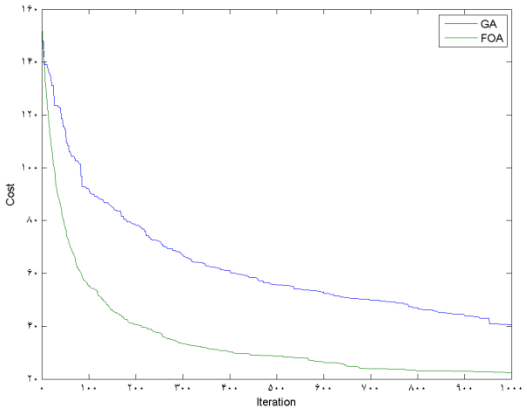
comp09



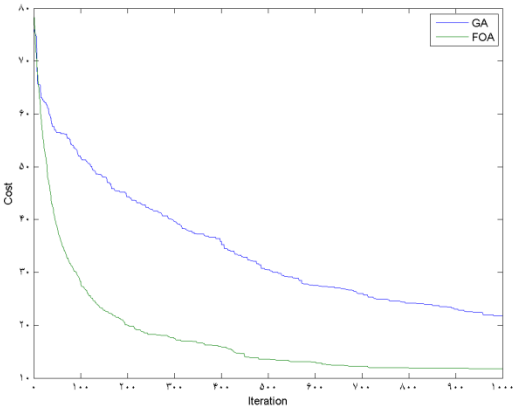
comp10



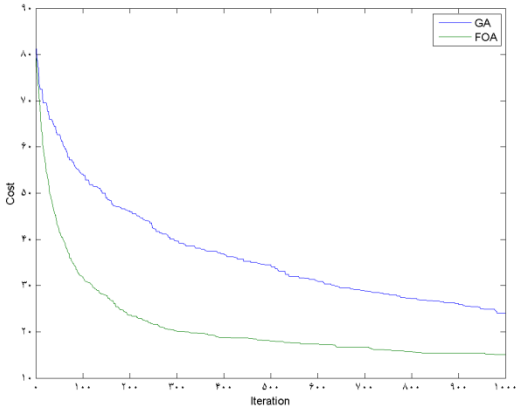
comp11



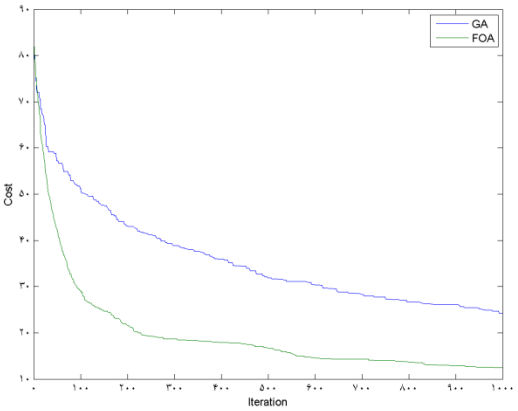
comp12



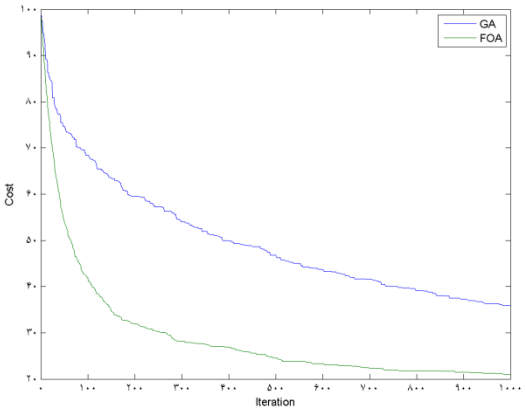
comp13



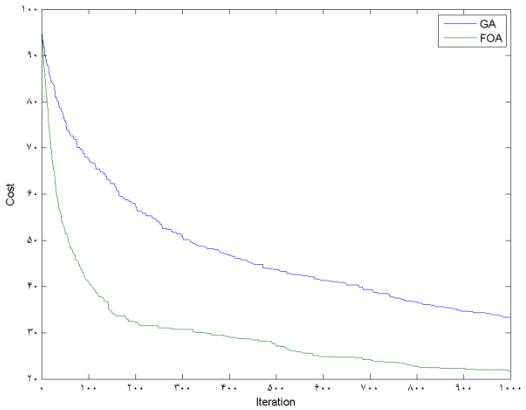
comp14



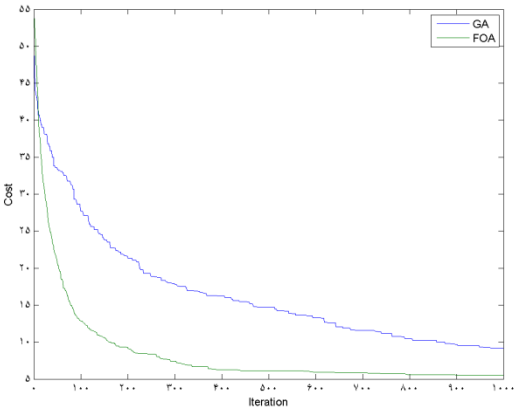
comp15



comp16



comp17



comp18

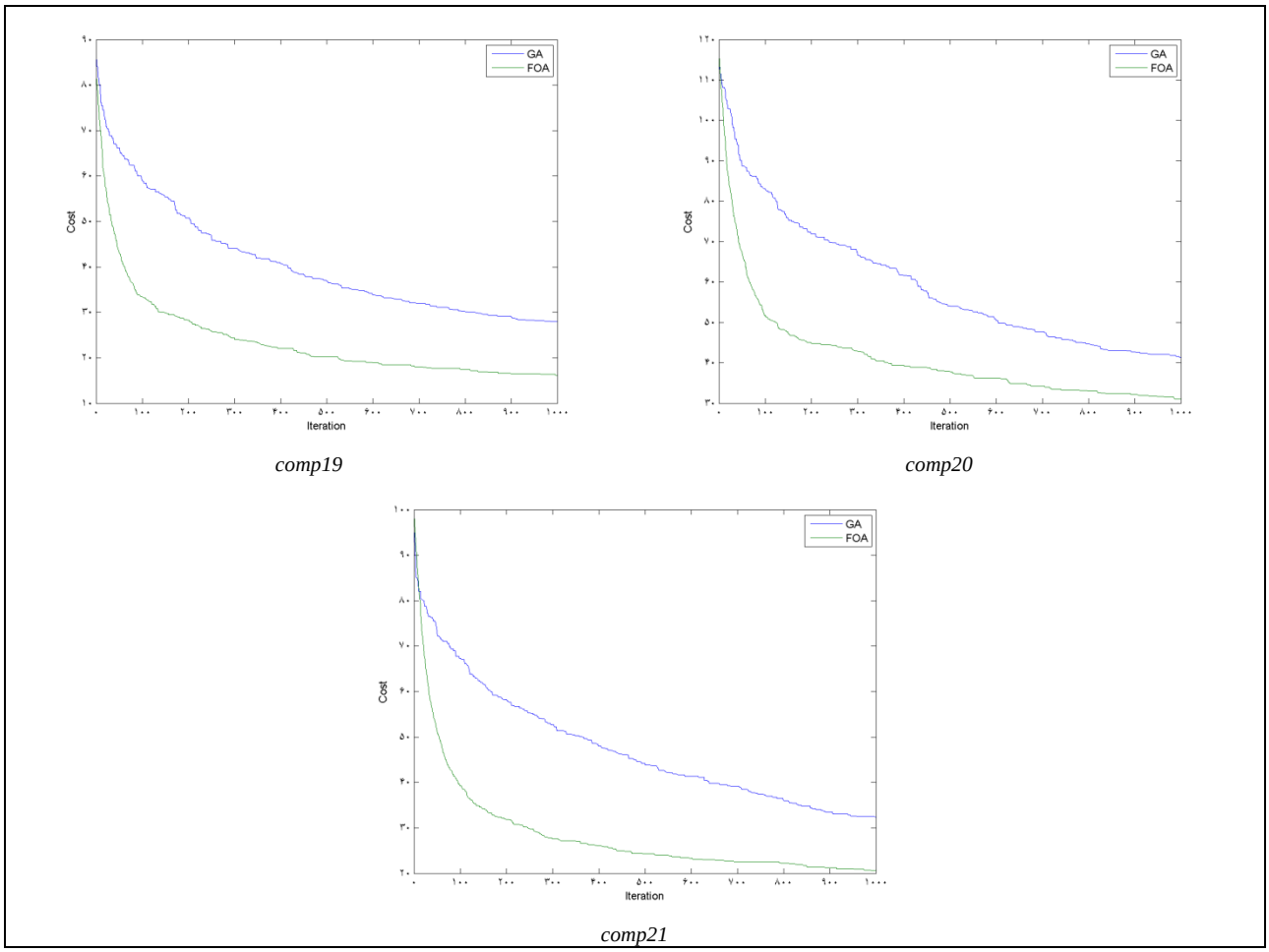


Figure 7: Comparison of Forest and Genetics algorithms for 21 different databases in ITC2007

Table 3 shows the weighted sum of unsatisfied constraints in the best result obtained until the 1000th iteration for Forest and Genetics algorithms in databases with different sizes and conditions. Runtime is also shown in the table. As is clear, compared to GA, the Forest algorithm in most cases (90.5%) had shorter runtime and achieved better results in less time.

Table 3: Cost Function and runtime of the Forest algorithm compared with GA until the 1000th iteration in databases with different sizes

Dataset	GA(Cost)	FOA(Cost)	GA(time)	FOA(time)
Comp1	**6.8000**	7.7500	52.4424	**4.0702**
Comp2	30.5000	**20.3500**	428.1049	**242.5067**
Comp3	29.1500	**14.4500**	398.1583	**187.3426**
Comp4	22.0500	**11.1000**	298.3329	**207.0348**
Comp5	30.0500	**16.1000**	423.4766	**196.4453**
Comp6	38.9000	**26.4500**	437.2634	**318.7111**
Comp7	49.9000	**36.2000**	**535.6721**	563.1129
Comp8	22.1000	**13.7500**	352.3408	**210.8135**
Comp9	26.3000	**18.1500**	432.6287	**237.6737**
Comp10	40.6500	**26.1000**	383.1883	**354.3523**
Comp11	6.1000	**5.8000**	51.1953	**6.6937**
Comp12	40.4500	**22.4500**	549.4061	**483.8322**
Comp13	21.8000	**11.8000**	283.8159	**245.0613**
Comp14	24.0500	**15.1000**	276.4695	**192.3106**

Comp15	24.2500	**12.5000**	276.9967	**175.8208**
Comp16	35.8500	**21.0500**	**348.5013**	411.2633
Comp17	33.3500	**21.8500**	859.9664	**214.2502**
Comp18	9.2000	**5.5000**	204.1438	**63.4649**
Comp19	27.9000	**16.1000**	293.6720	**162.2599**
Comp20	41.3500	**31.0000**	671.4165	**434.3658**
Comp21	32.4500	**20.5500**	382.3586	**271.9913**

Table 4: Comparison results of the Forest algorithm and GA with fixed cost function value in databases with different sizes

Dataset	GA(iteratin)	FOA(iteratin)	Cost
Comp1	**915**	978	7.2500
Comp2	934	**217**	31.0000
Comp3	962	**115**	29.5000
Comp4	905	**165**	22.3500
Comp5	957	**149**	30.5500
Comp6	959	**230**	39.3500
Comp7	961	**199**	50.4000
Comp8	889	**224**	22.6000
Comp9	930	**187**	26.8000
Comp10	983	**179**	41.1500
Comp11	907	**676**	6.6000
Comp12	953	**196**	40.9500
Comp13	959	**161**	22.3000
Comp14	980	**188**	24.5500
Comp15	979	**150**	24.7500
Comp16	946	**140**	36.3500
Comp17	978	**159**	33.8500
Comp18	903	**175**	9.7000
Comp19	914	**191**	28.4000
Comp20	990	**322**	41.8500
Comp21	940	**169**	32.9000

Table 4 shows the number of iterations where Forest and Genetics algorithms reached fixed precision. According to the results, in 95% of cases the Forest algorithm achieved optimal result in less time.

6. Conclusion

The Forest algorithm is an optimization method proposed in recent years by researchers for optimization problems. This paper presented how to use the algorithm for course timetabling as well as evaluation conditions, evaluation results and comparison with GA. The Forest algorithm was compared with GA for standard databases in different conditions. According to the results, in more than 99% of iterations, the Forest algorithm had lower unsatisfied constraints compared to GA, thus it had lower cost. The Forest algorithm also had shorter runtime in most cases and achieved better results in less time. Overall the evaluation results indicated higher accuracy and speed for the proposed method compared to GA.

7. Future works

So far many methods for course timetabling have been proposed and evaluated by various researchers. However, there is room for future works. Here are some methods that can be discussed for course timetabling.

- Considering the preferences of professors and students about time of courses in course timetabling
- Design of parallel algorithms for course timetabling with existing methods to increase efficiency

References

[1] Der-Fang, S.; A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences; Expert Syst. Appl. Volume 38, Pages 235-248, (January 2011).

[2] Qaurooni, D., Akbarzadeh, M.R.; Course timetabling using evolutionary operators; Applied Soft Computing; Volume 13, Issue 5, Pages 2504-2514, (2013).

[3] Asmuni, H., Burke, E.K., Garibaldi, J.M.; Fuzzy Multiple Heuristic Ordering for Course Timetabling; In: Proceedings of 5th UK Workshop on Computational Intelligence (UKCI '05), Pages 302–309, (2005).
[4] Thepphakorn, T., Hicks, Ch., Pongcharoen, P.; An Ant Colony Based Timetabling Tool; International Journal of Production Economics, Volume 149, Pages 131–144, (March 2014).
[5] Azmi Al-Betar, M., Tajudin Khader, A.; A harmony search algorithm for university course timetabling. Annals of Operations Research, Volume 194, Issue 1, Pages 3-31, (April 2012).
[6] Chen, R.-M.; Shih, H.-F. Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search. Algorithms 2013, 6, 227-244.
[7] Adrianto, D.; Comparision Using Particle Swarm Optimization and Genetic Algorithm for Timetable Scheduling; Journal of Computer Science 10 (2): 341-346, 2014.
[8] Zadeh L. A., Fuzzy Logic: Computing with Words, IEEE Transactions on Fuzzy Systems (TFS), vol. 4, no. 2, pp. 103-111, May 1996.
[9] Mamdani E. H., Advances in the Linguistic Synthesis of Fuzzy Controllers, International Journal of Man-machine Studies (IJMMS), vol. 7, no. 1, pp. 1-13, 1976.
[10] Azmi Al-Betar, M., Tajudin Khader, A.; University Course Timetabling Using a Hybrid Harmony Search Metaheuristic Algorithm; Applications and Reviews, VOL. 42, no. 5, September 2012.
[11] Ayob, M., Jaradat, G.; Hybrid Ant Colony Systems For Course Timetabling Problems. IEEE 2nd Conference on Data Mining and Optimization 27-28 October 2009, Selangor, Malaysia, 120-126, 2009.
[12] Babaei, H., Karimpour, J., Hadidi, A., A Survey of Approaches for University Course Timetabling Problem, Computers & Industrial Engineering 2014;
[13] Badoni, R. P., Gupta, D.K., Mishra, P.; A new hybrid algorithm for university course timetabling problem using events based on groupings of students; Computers & Industrial Engineering vol.78, pp. 12–25, 2014.
[14] Ozcan, E., Alkan, A.; A memetic algorithm for solving a timetabling problem: An incremental strategy; Proc. of the 3rd Multidisciplinary Int. Conf. On Scheduling: Theory and Applications; pp. 394-401, At Paris, France, 2014;
[15] Mayer, A., Nothegger, C., Chwatal, A., Raidl, G.; Solving the Post Enrolment Course Timetabling. Problem by Ant Colony Optimization. Annals of Operations Research; Vol. 194 Issue 1, p 325-339, Apr2012.
[16] Chmait, N., Challita, K.; Using Simulated Annealing and Ant-Colony Optimization Algorithms to Solve the Scheduling Problem; Computer Science and Information Technology 1(3): 208-224, 2013;
[17] Sabar, N. R., Masri, A., Kendall, G., Qu, R.; A honey-bee mating optimization algorithm for educational timetabling problems; European Journal of Operational Research 216 (2012) 533–543.
[18] M. Alzaqebah a,n, S. Abdullah; Hybrid bee colony optimization for examination timetabling problems; Computers & Operations Research 54 (2015) 142–154.
[19] Schaerf, A.; A Survey of Automated Timetabling; 1/02/1999; 16:48; no v.; p.3.
[20] Kristiansena, S., Stidsena , T. R.; A Comprehensive Study of Educational Timetabling a survey;
[21] Pillay, N., A survey of school timetabling research
[22] Fong, C.W., Asmuni, H.b., McCollum, B., McMullan, P., Omatu, S.; A new hybrid imperialist swarm-based optimization algorithm for university timetabling problems, Information Sciences (2014), doi: http://dx.doi.org/10.1016/j.ins.2014.05.039
[23] Afsari, F., Eftekhari, M., Zahedi, M.; Planning Timetabling University Cource Using Ant_fuzzy; 14th Annual Conference of computer society of iran, Tehran, Amirkabir University (2008).
[24] Ghaemi M, Feizi-Derakhshi, Forest optimization algorithm. Expert Systems with Applications, Volume 41, Pages: 6676–6687, 2014.
[25] http://tabu.diegm.uniud.it/ctt/index.ph