Hybridizing Genetic Algorithm with Biased Chance Local Search

Mehdi Basikhasteh, and Mohamad A. Movafaghpour

***Abstract*—**This paper explores university course timetabling problem. There are several characteristics that make scheduling and timetabling problems particularly difficult to solve: they have huge search spaces, they are often highly constrained, they require sophisticated solution representation schemes, and they usually require very time-consuming fitness evaluation routines. Thus standard evolutionary algorithms lack of efficiency to deal with them. In this paper we have proposed a memetic algorithm that incorporates the problem specific knowledge such that most of chromosomes generated are decoded into feasible solutions. Generating vast amount of feasible chromosomes makes the progress of search process possible in a time efficient manner. Experimental results exhibit the advantages of the developed Hybrid Genetic Algorithm than the standard Genetic Algorithm.

***Keywords*—**University Course Timetabling, Memetic Algorithm, Biased Chance Assignment, Optimization.

I. Introduction

THE University Course Timetabling Problem (UCTP) consists of scheduling a set of lectures for each course within a given number of rooms and time periods. In a UCTP, we assign an event (course-lecture) into a time slot and also assign a number of resources (professors, students, and rooms) in such a way that there is no conflict between the resources, time slots and events. Another similar problem is school timetabling problem (STP). The main difference between UCTP and the STP is that university courses can have common students, whereas school classes are disjoint sets of students. If two courses have common students then they conflict, and they cannot be scheduled at the same period. Moreover, school teachers always teach just one course, whereas in universities, a professor can teach a set of course. In addition, in the UCTP, availability of rooms (and their size) plays an important role, whereas in the STP they are often neglected because, in most cases, we can assume that each class has its own room. As mentioned by Carter and Laporte (1998) the UCTP is a multi-dimensional assignment problem, in which students and teachers (or faculty members) are assigned to courses, lectures or classes and events (individual meetings between students and teachers) are assigned to classrooms and time slots.

Several authors split the requirements into hard and soft ones (Eiselt and Laporte, 1987). The hard requirements are included in the constraints and they make the search space, whereas the soft ones are included in the objective function. Soft requirements generally include event spreading constraints and room capacity constraints [5]. The real world UCTP consists of different constraints: some are hard and some are soft. Hard constraints have a higher priority than soft. The objective of the UCTPs is to satisfy the hard constraints and to minimize the violation of the soft constraints.

Courses timetabling varies from university to university according to the resources and constraints. There is no known deterministic polynomial time algorithm for solving the UCTP. Because, Even et al. 1976 proved the UCTP is an NP-hard problem. So, it is very difficult to be solved by conventional methods and the amount of computation required finding optimal solution increases exponentially with problem size.

A wide variety of solution techniques and approaches for solving UCTPs have been described in the literature and evaluated by standard problem instances. Note that, there is a main difference between techniques and approaches; a technique is an algorithm or a set of algorithms for solving the problem (e.g., genetic algorithms). Instead an approach is a general framework for developing a solution algorithm (e.g., constraint logic programming). Burke and Petrovic (2002) classified these methods into four main types: sequential methods, clustering methods, constraint-based methods, and meta-heuristic methods.

Sequential methods order events using domain heuristics and then assign the events sequentially into valid time periods so that no events in the period are in conflict with each other [12]. In these methods, timetabling problems are usually represented as graphs where events (courses, lectures) are represented as vertices, while conflicts between the events are represented by edges (de Werra, 1985). In the clustering methods the set of events is split into some clusters which satisfy hard constraints and then the clusters are assigned to time periods to fulfill the soft constraints. Different optimization techniques have been employed to solve the problem of assigning the clusters of events into time periods (Balakrishnan et al. 1992). The main drawback of these approaches is that the clusters of events are formed and fixed at the beginning of the algorithm and that may result in a poor quality timetable. In the constraint-based methods a timetabling problem is modeled as a set of variables (i.e., events) to which values (i.e., resources such as rooms and time periods) have to be assigned to satisfy a number of constraints (Brailsford et al. 1999). Usually a number of rules

M. Basikhasteh is with the Mathematics Department of Islamic Azad University, Dezful Branch, Dezful, Iran (e-mail: basikhasteh@iaud.ac.ir)

M. A. Movafaghpour is with the Jundi Shapur University of Technology, Dezful, Iran. (Corresponding author, phone: +98-916-3419562; fax: +98-641-6266666; e-mail: movafaghpour@jsu.ac.ir).

are defined for assigning resources to events. When no rule is applicable to the current partial solution a backtracking is performed until a solution is found that satisfies all constraints. In the last two decades a variety of meta-heuristic approaches such as simulated annealing (SA), tabu search (TS), genetic algorithms (GAs) and hybrid approaches (e.g., memetic algorithms (MAs)) have been investigated for timetabling. Meta-heuristic methods begin with one or more initial solutions and employ search strategies that try to avoid local optimum. All of these search algorithms can produce high quality solutions but often have a considerable computational cost.

Meta-heuristics are divided into two categories, local search-based and population-based methods. The local search-based methods consider one solution at a time. The solution undergoes changes iteratively until a final solution which is usually in the same region of the search space as the initial solution is reached. They often use neighborhood structures guided by a given acceptance rule to improve the quality of solution. Although the biggest merit of using these methods is their strength of fine-tuning the solution more structurally and more quickly than population-based methods, the main drawback is that they have a tendency to get stuck in a small region of the search space. This is mainly due to local search-based methods focusing on exploitation rather than exploration, which means that they move in one direction without performing a wider scan of the entire search space (Al-Betar and Khader, 2010).

The hybridization method (an evolutionary algorithm together with a local search) has been given various other names in the literature such as memetic algorithms, hybrid genetic algorithms, genetic local search algorithms and etc (Hart et al. 2004). In this paper, a memetic algorithm is proposed for solving the UCTP, which combines a local search technique into GA. MAs are a class of meta-heuristic methods, which combine the population-based method GA, with local search made by individuals. Many researchers have applied MAs to address timetabling problems by combining GAs and local search techniques.

Burke and Newall (1999) proposed a multi stage evolutionary algorithm which integrated an evolutionary algorithm with a decomposition method. Real data sets were used to evaluate the efficiency of the algorithm. The results of real set of instances show the efficiency of their proposed algorithm. Abdullah et al. (2005) developed a Variable Neighborhood Search (VNS) approach which used a fixed tabu list to penalize particular neighborhood structures. The authors continue their work by developing a hybrid Evolutionary Algorithm with VNS for solving UCTP with very successful outcomes (Abdullah et al. 2007). As mentioned before, inserting local search within GA is considered as an effective way to produce high quality solution than using GAs. Abdullah and Turabieh (2008) applied a sequential search algorithm as a local search into GA to improve the timetable by reducing the number of soft constraint violated. They have applied repair process for rectifying infeasible chromosomes that were generated during evolution process. The repair function of their algorithm was able to change infeasible timetable to feasible one.

Jat and Yang (2008) proposed a memetic algorithm for UCTP, which integrates two local search techniques into GAs. The first local search technique was based on events (i.e. courses and subjects) and the second was based on time slots. They considered three soft constraints and the goal of UCTP was to minimize the soft constraint violations of a feasible solution. Both local search techniques work in two steps. In the first step a feasible solution was generated base on hard constraint violations. They defined a solution as a feasible solution if that satisfied all hard constraints. If there are hard constraint violations for either an event or a time slots, local searches try to resolve them by applying moves in the three neighborhoods structures until a termination condition was reached. In the second step, after reaching the state of a feasible solution, local searches then deals with soft constraints and again perform a similar process as in the first step on each event or time slot to reduce its soft constraint violations.

In this paper we have proposed a chance based selection step enhanced by a biased weighting process as a rapid local search module. This module is then embedded in a Genetic Algorithm which results an efficient Memetic Algorithm. Developed MA is implemented on a spread sheet for to solve real world problems. In the remaining of the paper first we define the real world problem dealt with in details in section 2, and then the developed algorithm is described in section 3. Some experimental results are provided in section 4 and some concluding remarks with threads for feature studies are reviewed in section 5.

II. University Course Timetabling In I.A.U. Dezful

Islamic Azad University Dezful branch (I.A.U. Dezful) enrolls about 18000 students in over 20 academic departments. The Department of Mathematical Sciences (DMS) offers about 400 hours of courses in each semester through the cooperation of about 45 full time and part time faculty staff. Each semester, all departments provide the DMS an estimated enrollment and special requirements (e.g., with regard to a special instructor, room, or timeslot) for each section of each course.

Because of the growing demand for higher education in recent years, DMS lacks of enough educational spaces to plan the courses as compared with last years; and timetabling process turns out to be a more difficult problem. The problem of course timetabling in DMS entails planning the courses specific to students of mathematics (specific courses) and courses serviced to the students of other departments (service courses). The scheduler at DMS must assign 150 classes to about 7 classrooms. The assignment has to take a number of objectives into consideration. A room with fewer seats than students is undesirable, as is a room that is much too large. In addition, the location of the room is also important. From a professor's point of view, it is nice to have a room that is close to his or her office. From a students' point of view it is convenient to have consecutive classes close together.

It is not easy to state a formal objective for this optimization problem, since there are often no clear priorities. For example, if there is no room to accommodate both Numerical Methods and General Math at the same time, then it is not easy to make a choice based on a priori principle. Fortunately, some policy guidelines had been established on standard time patterns for offering courses. The twelve-hour day, starting at 8 a.m., is divided into twelve one hour blocks. Classes may be scheduled only for single, dual or triplet consecutive blocks. The classes with more weekly teaching hours are to be departed into two partitions each requiring less than three consecutive time blocks.

Students of mathematics admitted in the same year all called 'year-mate'. Since the policy of DMS is to facilitate the process of course selection for student, they are categorized into groups called 'year-mates' and there is a recommended set of courses for each set of year-mates designated for each semester. The hard constraint 'no student conflict' compels the planner to avoid the overlaps between the schedules of courses gathered in a same recommended courses list.

Each instructor is capable of teaching a set of courses and is favored to have its working hours in certain timeslots in distinct days. No instructor conflict is allowed, i.e. each instructor can teach at most one section of a course at a time. Each course has a distinct 'teaching units' such as: 2, 3, …, 6. Which it means that course is to be taught 2, 3, …, 6 hours a week respectively. No course can be planned to be taught continuously for more than 3 hours. In other words the courses with 4 or more 'teaching units' should be divided into two sessions during the week and the sessions can be held in different rooms. Beside the standard constraints usually dealt with in a university course timetabling, DMS faces some other hard constraint which makes the timetabling more complicated. They are listed below:

- No classroom conflict: Available classrooms are bounded to be at most 7 classrooms a day and are extendable to 8 in special cases. No overlap is allowed in the schedule of the classrooms.
- No student conflict: a group a year-mates with the same recommended courses list should have the possibility to select the recommended courses in each semester without any conflict. This means that each course in each recommended courses list for a given semester should have at least one of its sections planned so that it has no time overlap with all other courses of that list.
- No instructor conflict: each instructor prefers to teach some arbitrary time blocks through the week. All the sections assigned to each instructor should be planned through his/her preferred time blocks with no overlaps.
- Each full time mentor has to teach at least 16 hours a week.
- Each full time assistant professor has to teach at least 9 hours a week.
- Each faculty member (part time or full time) has to teach at most 23 hours a week.
- Working days of each full time faculty member has to be at least 4 days a week.
- Each working day consists of twelve one hour time blocks and the 5th one is reserved for lunch and prayers time. All sessions planned before 5th time block have to be finished before lunch time.
- Courses should be planned for 2 or 3 hour sessions (each session is 2 or 3 consecutive time blocks in a day) through the week and those having 4, 5, or 6 teaching units should be departed into two partitions each with 2 or 3 hour sessions.

DMS faces the soft constraints listed below:

- Decrease the distance traveled by each instructor as much as possible. Since there may be some specific or service courses assigned to an instructor, the rooms available to planner are scattered in different departments physically distributed in the university.
- Increase the efficiency of the timetable for each professor. Lessen the idle slots between a pair of working slots.
- Increase the efficiency of the timetable of each room. Lessen the idle interim single slots.
- Increase the efficiency of the timetable for each student. i.e. Decrease the active days required for each set of recommended courses list.

III. MEMETIC ALGORITHM

It is possible to think of a memetic algorithm as an evolutionary algorithm that incorporates knowledge about the problem domain being solved (Burke and Landa Silva, 2005). This knowledge can be in the form of specialized operators, heuristics and other local searches that contribute towards self-improvement ability in the individuals of the population. Since memetic algorithms also are known as a combination of genetic algorithms with local search heuristics, they are also called genetic local search, hybrid genetic algorithms, hybrid evolutionary algorithms (Talbi, 2002). This type of hybrid approach has been applied to a vast number of optimization problems with considerable success.

It is generally believed that memetic algorithms are successful because they combine the explorative search ability of genetic algorithms and the exploitive search ability of local searches. An analogy is that the evolutionary part of a memetic algorithm attempts to simulate the genetic evolution of individuals through generations, while the local search part attempts to simulate the individual learning within a lifetime. This local search can be for example, constructive heuristics, repair methods, specialized self-improvement operators, etc. The local search phase can be applied before, after or in between the genetic operations (Fig. 1). Krasnogor (2002) argues that in a truly memetic system:

1. Memes also evolve representing the way in which "individuals learn, adopt or imitate certain memes or modify other memes" and,

2. The distribution of memes changes dynamically within the population representing the effects of "teaching, preaching, etc." within the population of individuals.

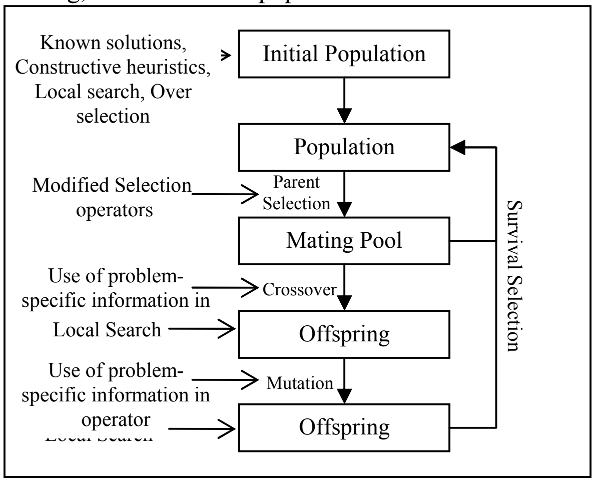
Known solutions, Constructive heuristics, Local search, Over selection
Initial Population
Population
Modified Selection operators
Parent Selection
Mating Pool
Use of problem-specific information in
Crossover
Local Search
Offspring
Use of problem-specific information in operator
Mutation
Offspring
Survival Selection

Fig. 1 Possible places to incorporate knowledge or other operators within MA (Hart et *al*. 2004)

In this paper we have introduced a biased decoding scheme for MA and developing Biased Decoded Memetic Algorithm (BDMA) in which a string of random numbers is decoded into a solution (timetable). We have applied the problem specific knowledge to assign biased chance to each component (course, professor, room, day, time slot) of the chromosomes in order to avoid infeasible timetables satisfying hard constraints.

IV. Developed Memetic Algorithm

In order to solve the problem of UCTP in DMS we have developed 'Biased Decoded Memetic Algorithm' (BiD-MA). BiD-MA takes the following parameters as input:

- Properties of the courses: teaching units and the number of sections required of each course,
- Teaching interests: the ability or interests of professors to teach the courses,
- Time interests: the time blocks afforded by each instructor as teaching time,
- Recommended courses list: the courses recommended to be taken in the same semester by each set of year-mates,

and tries to produce feasible timetables satisfying hard constraints. Soft constraints are considered as evaluation function in selection phase. Five evolutionary operators of the BiD-MA are discussed below.

A. Encoding Scheme

Every chromosome consists of a main string and a tail string. The main string is a sequence of 5-tuple sub-strings and the tail string is a sequence of triplet sub-strings. Each 5-tuple sub-string determines how to select a set of 1) course, 2) professor, 3) classroom, 4) day and 5) time slot to accommodate one section of a course and triplet sub-strings determine how to select a 1) classroom, 2) day, and 3) time slot to accommodate the second partition of the classes with teaching time more than 4 hours a week (Fig. 2).

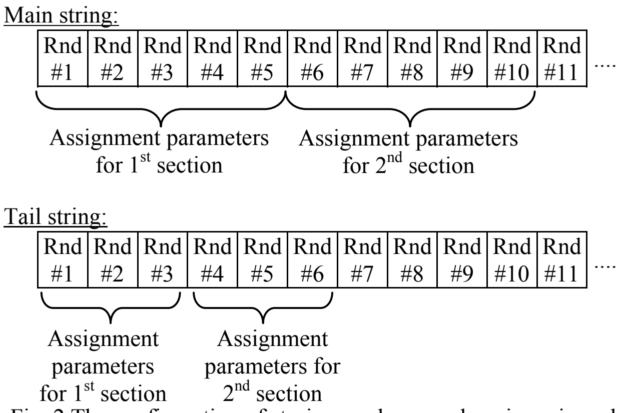

Fig. 2 The configuration of storing random numbers in main and tail string of a chromosome

B. Parent Selection

In order to form pair of parents, each parent is randomly selected from the population. Selected parents are removed from the population and then are moved to mating pool. Offsprings and parents are then evaluated based on their evaluation function and the next population is selected among them.

C. Crossover

Each pair of parents yields two offsprings. A binary mask string is generated to determine which genes should be copied from the first parent and which genes should be copied from the second one to reproduce each offspring. The mechanism of binary mask approach in illustrated in Fig. 3.

D. Mutation

Three percent of offsprings generated by crossover operator are randomly selected and passed to mutation operator.

E. Fitness Evaluation

In order to evaluate the fitness of each chromosome, it is decoded to its respective timetable satisfying hard constraints. The evaluation function is calculated as the violation of soft constraints in the schedule of professors, students, and classrooms is measured in form of the summation of idle time blocks surrounded by two working blocks in a daily schedule.

Parent #1:

a	b	c	d	e	f	g	h	i	j	k	L	m	n

Parent #2:

A	B	C	D	E	F	G	H	I	J	K	L	M	N

Binary mask:

1	1	1	1	0	0	0	1	1	1	1	1	0	0

Child #1:

a	b	c	d	E	F	G	h	i	j	k	L	M	N

Child #2:

A	B	C	D	e	f	g	H	I	J	K	L	m	n

Fig. 3 Applying a binary mask on parent #1 and parent #2 produces child #1 and child #2

```
1)  Read    the main sub-string and consider it as active sub-string
2)  Assign    weights to courses
3)  Read    the first random number stored in the first gene of active
    sub-string
4)  Select    a course based on roulette wheel
5)  If    the course requires 3 or less teaching hours a week
6)        "required time blocks" ← "teaching hours"
7)  Else
8)        "required time blocks for first partition of the course" ←
    roundup(("teaching hours")/2)
9)  End

10) Assign    weights to professors
11) Read    the next random number stored in the active sub-string
12) Select    a professor based on roulette wheel

13) Assign    weights to classroom
14) Read    the next random number stored in the active sub-string
15) Select    a classroom based on roulette wheel

16) Assign    weights to days
17) Read    the next random number stored in the active sub-string
18) Select    a day based on roulette wheel

19) Assign    weights to time blocks
20) Read    the next random number stored in the active sub-string
21) Select    a time block as the starting time block based on roulette
    wheel

22) If    the course requires 4 or more teaching hours a week
23)       "required time blocks for second partition of the course" ←
    ("teaching hours" – "required time blocks for first partition of the
    course")
24) Else
25)       Stop
26) End
27) Read the tail sub-string and consider it as active sub-string
28) Repeat 13-21 for the second partition of the course.
29) Stop
```

Fig. 4 Enrolling an event based on the random numbers stored in chromosomes

Events are enrolled consecutively by reading consecutive sub-strings in main string and tail string of each chromosome. Each 5 random number stored in the main sub-string are used in a roulette wheel procedure to select: course, professor, classroom, day, and starting time block for single partitioned courses. the courses with 4 or more weekly teaching hours that have to fragmented into two partitions, the first partition is enrolled with regard to the random numbers stored in the main sub-string and the recourses for the second partition (classroom, day, starting time block) is enrolled by reading the random numbers stored in the tail sub-string.

As it was mentioned before a ‘Biased Decoding’ procedure is developed in this paper. This procedure enrolls each individual event based on hierarchically assigning a biased chance to the candidate alternatives for each of 5 components of an event (course, professor, classroom, day, and starting time block). And then the random number stored in the respective gene in the related sub-string of the chromosome is used in a roulette wheel procedure to select the best alternative for each component. Decoding procedure is illustrated in Fig. 4.

In steps 2, 10, 13, 16, and 19 if the weighting modules assign uniform chance to all alternatives, no problem specific knowledge is applied, but we have proposed biased weighting in which more promising alternatives take more chance to be selected. This issue increases the probability of generating more probable feasible timetables. Each of weighting modules named above uses the respective considerations below to increase the probability of reaching feasible timetables.

- Weighting courses: The group of specific courses is highly prioritized than the group of service courses. Within each group, the courses with less candidate feasible professors and less required sections are given more weight.
- Weighting professors: For the selected course, professors with more free and feasible time blocks and less number of favorite courses are given more weight.
- Weighting days: For the selected course and professor the days with more feasible options for starting time block in different classrooms are given more weight.
- Weighting starting time blocks: For the selected course, professor, and day, the starting time blocks whose lessen the evaluation function in all possible classrooms are given more weight.
- Weighting classrooms: For the selected course, and professor, day, and starting time block, the classrooms whose lessen the evaluation function are given more weight.

The term "feasible options" refers to satisfying hard constraints listed in section 2. As it was mentioned in section 4-5 the evaluation function of each chromosome (timetable) is measured in form of the summation of idle time blocks surrounded by two working blocks in a daily schedule of professor, classroom and each group of year-mates. When assigning weights to time block in order to select the starting time block of each event, the Evaluation Function (EF) of the decoded chromosome is also calculated. This EF should be minimized.

F. Survival Selection

The fitness of *i*'th chromosome with EF_i is calculated as below:

$$Fitness_i = 1 \div (1 + EF_i).$$

Fittest chromosomes are selected by a roulette wheel based on the chromosomes' fitness calculated above. Elite size was set on 1 chromosome.

V. Experimental Results

In order to analyze the capabilities of developed BiD-MA, a set of real world data for timetabling in the Department of Mathematical Sciences of I.A.U. Dezful was gathered. The problem consists of planning for 47 instructors and about 50 courses (30 specific courses and 20 service courses) summing up to 117 sections or equivalently about 400 hours of course work. The instances were produced by selecting sub sets of this dataset by varying number classrooms, total sections required, and the number of professors.

Developed algorithm is compared with the basic GA in which the decoding procedure is actuated based on uniform weights. BiD-MA and GA are coded in MATLAB 7.0 and executed on a PC with 256 MB of RAM and a 1.6 GHz CPU. Total generation for GA and BiD-MA were bounded to 3 generations with a set of 10 chromosomes.

As it can be inferred from table 1 uniform weighting (used in GA) can not yield feasible solutions to the problem but biased weighting in BiD-MA produces feasible schedules.

VI. Concluding Remarks

In this paper we focused on the problem of university course timetabling and developed a Memetic Algorithm with improved decoding approach. We proposed a novel approach to incorporate problem specific knowledge into the standard MA. The approach consists of biasing the weight of alternative sources to be selected and forming a complete timetable. The main advantage of this approach is providing a high speed search with an efficient combination of diversification and intensification in the search process. Since producing feasible timetables for UCTP is difficult, we have proposed a biased decoding procedure in which the promising alternatives are charged with greater weights to be selected at a higher chance in a roulette wheel procedure.

Acknowledgment

This research was supported by I.A.U. Dezful. We also express our gratitude to the Department of Mathematical Sciences of I.A.U Dezful for the supply of the data used to analyzing the algorithm developed in this paper.

References

[1] Abdullah S. and H. Turabieh, (2008). Generating University Course Timetable Using Genetic Algorithms and Local Search. Third 2008 International Conference on Convergence and Hybrid Information Technology.

[2] Abdullah, S., E. K. Burke, B. McCollum, (2005). An Investigation of Variable Neighborhood Search for Course Timetabling. In: The Proceedings of the 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), New York, USA, July 18 -21, pp. 413–427.

[3] Abdullah, S., E. K. Burke, B. McCollum, (2007). A hybrid evolutionary approach to the university course timetabling problem. IEEE congress on evolutionary computation 2007, pp. 1764–1768, Singapore.

[4] Al-Betar, M. A. and A. T. Khader, (2010). A harmony search algorithm for university course timetabling. Annals of Operations Research, In press.

[5] Aubin, J. and J. A. Ferland, (1989). A Large Scale Timetabling Problem. Computers and Operational Research 16(1): 67–77.

[6] Balakrishnan, N., Lucena, A., Wong, R.T., (1992). Scheduling examinations to reduce second-order conflicts. Computers and Operations Research 19, 353–361.

[7] Brailsford, S. C., C. N. Potts, B. M. Smith, (1999). Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research 119, 557–581.

[8] Burke E. K. and J. D. Landa Silva, (2005). The Design of Memetic Algorithms for Scheduling and Timetabling Problems, in "Recent Advances in Memetic Algorithms", Springer Berlin / Heidelberg, 289-311.

[9] Burke E. K. and J. P. Newall, (1999). A Multi-Stage Evolutionary Algorithm for Timetable Problem, IEEE Transactions in Evolutionary Computation 3(1), 63-74.

[10] Burke, E. K. and S. Petrovic, (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140(2): 266-280.

[11] Carter, M. W. and G. Laporte, (1998). Recent developments in practical course timetabling. Proc of the 2nd Int Conf on Practice and Theory of Automated Timetabling, LNCS 1408, pp. 3–19.

[12] Carter, M. W., (1986). A survey of practical applications of examination timetabling algorithms. Operations Research 34, 193–202.

[13] de Werra, D., (1985). An introduction to timetabling. European Journal of Operational Research 19, 151–162.

[14] Eiselt, H. A. and G. Laporte, (1987). Combinatorial Optimization Problems with Soft and Hard Requirements. Journal of the Operational Research Society 38: 785–795.

[15] Even, S., A. Itai, and A. Shamir, (1976). On the complexity of timetable and multicommodity flow problems. SIAM Journal on Computing, 5(4), 691–703.

[16] Hart, W. E., N. Krasnogor and J. E. Smith, (2004). Memetic Evolutionary Algorithms, Recent Advances in Memetic Algorithms: Studies in Fuzziness and Soft Computing, Springer-Verlag, 3-27.

[17] Jat, S. N. and S. Yang, (2008). A Memetic Algorithm for the University Course Timetabling Problem. 20th IEEE International Conference on Tools with Artificial Intelligence.

[18] Krasnogor N., (2002). Studies on the theory and design space of memetic algorithms. PhD thesis, Faculty of computing, engineering and mathematical sciences, University of the West of England, UK.

[19] Rossi-Doria O., M. Sampels, M. Birattari, M. Chiarandini, M. Dorigo, L. Gambardella, J. Knowles, M. Manfrin, M. Mastrolilli, B. Paechter, L. Paquete, and T. Stutzle, (2002). A comparison of the performance of different metaheuristics on the timetabling problem. Lecture Notes in Computer Science 2740, pp. 329–351.

[20] Talbi, E. G., (2002). A Taxonomy of hybrid metaheuristics. Journal of heuristics, 8, 541-564.

Mehdi Basikhasteh is a faculty member at Islamic Azad University of Dezful, Dezful, Iran. He received a B.S. degree in Statistics from Isfahan University of Technology and a M.S. degree in actuary from Shahid Beheshti University, Tehran. His research is focused on the application of statistical methods in swarm intelligence.

Mohamad Ali Movafaghpour is a senior PhD student at the Faculty of Engineering, Tarbiat Modares University, Tehran, Iran. He received a B.S. degree from Isfahan University of Technology, and an M.S. degree from Amirkabir University of Technology, both in industrial engineering. He is currently also an invited instructor at Jundi Shapur University of Technology. His research is focused on hybrid mathematical programming, Metaheuristic methods in Robotic Path Planning.